HelloWorld!

based on pinetime 0.11, upgrade will come soon

Create src/displayapp/screens/HelloWorld.h:

```
#pragma once

#include <lvgl/src/lv_core/lv_obj.h>
#include <cstdint>
#include "Screen.h"

namespace Pinetime {
  namespace Applications {
    namespace Screens {
      class HelloWorld : public Screen {
        public:
          HelloWorld(DisplayApp* app);
          ~HelloWorld() override;
          bool Refresh() override;
          bool OnButtonPushed() override;
          bool OnTouchEvent(TouchEvents event) override;

        private:
          bool running = true;
          lv_obj_t * hello_world_label;
      };
    }
  }
}
```

Create

src/displayapp/screens/HelloWorld.cpp:

```
#include "HelloWorld.h"
#include <lvgl/lvgl.h>

using namespace Pinetime::Applications::Screens;



HelloWorld::HelloWorld(Pinetime::Applications::DisplayApp *app) : Screen(app) {

  hello_world_label = lv_label_create(lv_scr_act(), nullptr);
  lv_label_set_text(hello_world_label, "Hello World ! ");
  lv_obj_set_auto_realign(hello_world_label, true);
  lv_obj_align(hello_world_label, nullptr, LV_ALIGN_CENTER, 0, 0);
}

HelloWorld::~HelloWorld() {
  lv_obj_clean(lv_scr_act());
}

bool HelloWorld::Refresh() {
  return running;
}

bool HelloWorld::OnButtonPushed() {
  running = false;
  return true;
}

bool HelloWorld::OnTouchEvent(Pinetime::Applications::TouchEvents event) {
  switch(event) {
    case TouchEvents::SwipeLeft:
    case TouchEvents::SwipeRight:
    default:
      return false;
  }
}
```

Edit src/displayapp/Apps.h :

Add HelloWolrd to the Enum:

```
#pragma once

namespace Pinetime {
  namespace Applications {
    enum class Apps {None, Launcher, Clock, SysInfo, Meter, Gauge, Brightness, Music,
FirmwareValidation, Paint, Paddle, Notifications, Twos, HeartRate, Navigation, HelloWorld};
  }
}
```

Edit src/displayapp/DisplayApp.cpp :

- Add

  ```
  #include "displayapp/screens/HelloWorld.h"
  ```

- Add Case Apps::HelloWorld to DisplayApp::RunningState()

```


void DisplayApp::RunningState() {
//  clockScreen.SetCurrentDateTime(dateTimeController.CurrentDateTime());

  if(!currentScreen->Refresh()) {
    currentScreen.reset(nullptr);
    lvgl.SetFullRefresh(Components::LittleVgl::FullRefreshDirections::Up);
    onClockApp = false;
    switch(nextApp) {
      case Apps::None:
      case Apps::Launcher: currentScreen.reset(new Screens::ApplicationList(this));
break;
      case Apps::Clock:
        currentScreen.reset(new Screens::Clock(this, dateTimeController, batteryController,
bleController, notificationManager, heartRateController));
        onClockApp = true;
```

```
        break;
//      case Apps::Test: currentScreen.reset(new Screens::Message(this)); break;
      case Apps::SysInfo: currentScreen.reset(new Screens::SystemInfo(this,
dateTimeController, batteryController, brightnessController, bleController, watchdog));
break;
      case Apps::Meter: currentScreen.reset(new Screens::Meter(this)); break;
      case Apps::Twos: currentScreen.reset(new Screens::Twos(this)); break;
      case Apps::Gauge: currentScreen.reset(new Screens::Gauge(this)); break;
      case Apps::Paint: currentScreen.reset(new Screens::InfiniPaint(this, lvgl));
break;
      case Apps::Paddle: currentScreen.reset(new Screens::Paddle(this, lvgl)); break;
      case Apps::Brightness : currentScreen.reset(new Screens::Brightness(this,
brightnessController)); break;
      case Apps::Music : currentScreen.reset(new Screens::Music(this,
systemTask.nimble().music())); break;
      case Apps::Navigation : currentScreen.reset(new Screens::Navigation(this,
systemTask.nimble().navigation())); break;
      case Apps::FirmwareValidation: currentScreen.reset(new
Screens::FirmwareValidation(this, validator)); break;
      case Apps::Notifications: currentScreen.reset(new Screens::Notifications(this,
notificationManager, Screens::Notifications::Modes::Normal)); break;
      case Apps::HeartRate: currentScreen.reset(new Screens::HeartRate(this,
heartRateController)); break;
      case Apps::HelloWorld: currentScreen.reset(new Screens::HelloWorld(this)); break;
    }
    nextApp = Apps::None;
  }
  lv_task_handler();
}
```

Edit: src/displayapp/screens/ApplicationList.h:

- Increment ScreenList size
- Uncomment std::unique_ptr... CreateScreen3();

```
#pragma once
```

```
#include <memory>

#include "Screen.h"
#include "ScreenList.h"

namespace Pinetime {
  namespace Applications {
    namespace Screens {
      class ApplicationList : public Screen {
        public:
          explicit ApplicationList(DisplayApp* app);
          ~ApplicationList() override;
          bool Refresh() override;
          bool OnButtonPushed() override;
          bool OnTouchEvent(TouchEvents event) override;
        private:
          bool running = true;

          ScreenList<3> screens;
          std::unique_ptr<Screen> CreateScreen1();
          std::unique_ptr<Screen> CreateScreen2();
          std::unique_ptr<Screen> CreateScreen3();
      };
    }
  }
}
```

Edit: src/displayapp/screens/ApplicationList.cpp:

Uncomment the createScreen3

```
ApplicationList::ApplicationList(Pinetime::Applications::DisplayApp *app) :
        Screen(app),
        screens{app, {
                [this]() -> std::unique_ptr<Screen> { return CreateScreen1();
},
                [this]() -> std::unique_ptr<Screen> { return CreateScreen2();
```

```
},
                [this]() -> std::unique_ptr<Screen> { return CreateScreen3(); }
          }
        } {}
```

Edit CreateScreen3, edit Apps::HelloWorld

```
std::unique_ptr<Screen> ApplicationList::CreateScreen3() {
  std::array<Screens::Tile::Applications, 6> applications {
          {{"H", Apps::HelloWorld},
           {"B", Apps::Gauge},
           {"C", Apps::Clock},
           {"D", Apps::Music},
           {"E", Apps::SysInfo},
           {"F", Apps::Brightness}
          }
  };
```

Edit CMakeLists.txt:

- Add .cpp to list(APPEND SOURCE_FILES
- Add .h to set(INCLUDE_FILES

```
list(APPEND SOURCE_FILES
...
        displayapp/screens/HelloWorld.cpp
...

set(INCLUDE_FILES
...
        displayapp/screens/HelloWorld.h
 ...
```

Compile

```
docker run --rm -it -v $(pwd):/sources pfeerick/infinitime-build
```

DFU is then stored in `build/output`

Revision #12

Created Thu, Feb 4, 2021 8:38 PM by Romain

Updated Wed, Mar 10, 2021 8:33 PM by Sylvain