

Architektúra

Elvek

A szoftver architektúrájának tervezésekor fő szempontok voltak a modularitás, hordozhatóság és a teljesítmény. Nem elhanyagolható igény továbbá a kis méret sem. Mindezek teljesítése végett több alkalommal ismerhető fel az objektum-orientált szemlélet.

Ezek miatt a programban nincs globálisan elérhető állapot, ilyen csakis a megfelelő komponens függvényének meghívásával lehet elérni.

Hardverközeli

Madártávlatból három fő részre osztható a kódbázis. Legalul helyezkedik el a *hardware-access*, ami a legalapvetőbb hardveres funkciókat látja el egy kényelmesen használható API-val. A következő hardver funkciókért felelős modulok találhatók itt.

- EEPROM (interfész blokkoló, belső működés megszakítás alapú)
- Energiahatékonyság
- SPI (sebesség végett blokkoló működés)
- Időzítés (megszakítás alapú működés)
- Digitális kimenet

Másik mikrokontrollerre váltás esetén a változtatások túlnyomó része az előbbi elemekben várható.

Ez a réteg rendelkezik egy Façade jellegű interfésszel, aminek két felelőssége van. Egyrészt deklarálja a hardware-access által biztosított funkciókat. Másrészt a teljes hardveres inicializálásért felel. Utóbbira a kis méret és az egyszerű használat miatt van szükség. A program elején inicializáljuk a HAL-t, utána pedig bárhol includeolva a megfelelő header-t, a teljes hardver funkcionalitást elérjük.

A következő réteg a *driver réteg*. Az itteni modulok lazán csatoltak és egyetlen közös tulajdonságuk, hogy a HAL-t használják funkcionalitásuk megvalósításához. A következő komponensek helyezkednek el itt.

- Kijelző
- Infra (megszakítás alapú működés)
- Perzisztens tároló
- Alvás ütemezés
- UART (megszakítás alapú működés)

Ezek többnyire egy egyszerű interfész (callback függvény) átadásával inicializálhatók. Erre a laza csatoltság miatt van szükség, igazából ez a dependency inversion principle C nyelvi megvalósítása.

A legmagasabb szintű funkciók az *alkalmazás réteg*-ben találhatóak. Maga a játék és annak logikája itt kerül megvalósításra. Ez saját belső architektúrával rendelkezik, amit a későbbiekben fogok kifejteni.

Végül egy fontos komponens maradt még. A *mediator* az alkalmazás és a driverek között helyezkedik el. Biztosítja, hogy az előbbi két rétegen belül és azok közt megvalósulhasson a kommunikáció. Felelőssége a komponensek orkesztrációja. Mindezt a megfelelő fejlécű függvények átadásával éri el a megfelelő inicializáló függvényhívásokban.

Alkalmazáslogika

A fő szempont a magasszintű kód írásának lehetővé tétele és a kompakt méret. Ezért a játékobjektumok közt egy prototípus alapú öröklési rendszer került kialakításra. Minden típus rendelkezik egy prototípussal, ami a méretét, a rajzoló és a frissítő függvényét tartalmazza. Maguk a prototípusok a programmemóriában kerülnek tárolásra. Az objektumok rendelkeznek egy mutatóval a prototípusukra, egy pozícióval és egy union-nal, amiben a típus specifikus állapotukat tárolják.

Az objektumok tárolására az objektumtároló szolgál. Ez igazából egy heterogén kollekció pár hasznos függvénnyel. Például segítségével lekérdezhetők az objektumok típus alapján vagy az alapján, hogy adott pixel pozíciót tartalmazzák-e. Továbbá az objektumok frissítését és rajzolását lehet innen kezdeményezni.

Az időközönkénti új objektumok létrehozását az esemény generátor kezdeményezi. Emellett a játékmenetért (indítás, újraindításért) felelős komponens is ebben a rétegben kerül megvalósításra.

Az összes játékobjektum típus rendelkezik saját komponenssel. Adott feladatukat a program többi részét használva igen magas szintű kóddal lehet megvalósítani.

Önálló komponense van továbbá az AI-nak, ami az élő felhasználó által használható interfészen keresztül irányíthatja a karaktereket.

Megemlíthető, hogy a kirajzolásra szánt spritesheetek is itt kerülnek definiálásra a kijelző által elvárt speciális formátumban. Ezek az EEPROM-ban kerülnek tárolásra.

Segédfunkciók

Egyéb sztenderd könyvtár jellegű segédfunkciók is helyet kapnak a kódbasisban. Ezek működése elég magától értetődő. Felsorolás jelleggel a következő komponensekről van szó.

- Véletlenszámgenerátor
- Vektor (2 dimenziós)
- Téglalap
- Makrók
 - Null
 - Bit műveletek