

Część 5 – właściwości obiektów

Aktywność obszarów

Obszary mogą być aktywne lub nieaktywne.

Zdefiniujemy w pomieszczeniu obszar, który odpowiada za zegar stojący na szafie, na razie bez akcji, które można na nim wykonać:

areas_def.h:

```
enum {  
    OBJECT_LIVING_ROOM,  
    OBJECT_LIVING_ROOM_CLOCK,  
    AREA_LAST_AREA  
};
```

Preferuję nazywanie obiektów i akcji według pomieszczeń, w których się znajdują, ze względu na czytelność przy większej ich liczbie.

areas_def.c:

```
// OBJECT_LIVING_ROOM_CLOCK  
{ 25,23, 15,17},
```

rooms_def.h

```
// ROOM_LIVING_ROOM  
{  
    // Actions  
    room_living_room_on_draw, // on room draw  
    ACTION_NONE, // on room enter  
    ACTION_NONE, // on room exit  
    // Areas. The main room area should be the last.  
    {  
        OBJECT_LIVING_ROOM_CLOCK,  
        OBJECT_LIVING_ROOM,  
    }  
},
```

Dodajemy obszar zegara **przed** obszarem całego ekranu, ponieważ sprawdzane są one w kolejności definicji i obszar większy zakryłby całkiem powierzchnię zegara.

objects_def.c:

```
// OBJECT_LIVING_ROOM_CLOCK  
{  
    // Name, variable  
    "Clock", AREA_INACTIVE,  
    // Actions on object  
    {  
        {TEXT_NONE, ACTION_NONE},  
    }  
},
```

Na początku ten obszar jest nieaktywny (AREA_INACTIVE). Sprawmy, że wywołanie polecenia "Examine" na wcześniej zdefiniowanym obiekcie (OBJECT_LIVING_ROOM) spowoduje dostrzeżenie zegara i uaktywnienie go. W tym celu zmodyfikujemy akcję

object_living_room_examine:

actions_def.c:

```
void object_living_room_examine(void)
{
    console_print("An old fashioned living room.",true);
    if(!area_activity_check(OBJECT_LIVING_ROOM_CLOCK))
    {
        console_print("Wow, there is a clock on the cabinet!",true);
        area_activity_on(OBJECT_LIVING_ROOM_CLOCK);
    }
}
```

W języku C wykrzyknik przy sprawdzaniu warunków to zaprzeczenie. Możliwe byłoby porównanie:

```
if (area_activity_check(OBJECT_LIVING_ROOM_CLOCK)==false)
```

ale sposób sprawdzania z wykrzyknikiem wygeneruje bardziej zwężły kod.

Teraz po “examine” pomieszczenia zostanie uaktywniony obszar zegara.



Właściwości obiektów

Każdy obiekt może przyjąć wartość od 0-127 lub kilka wartości jako flagi bitowe. Przetestujmy to na przykładzie zegara, który przy kolejnym sprawdzeniu czasu będzie pokazywał kolejną godzinę, zaczynając od godziny 14.

W tym celu zmodyfikujmy definicję zegara:

objects_def.c:

```
// OBJECT_LIVING_ROOM_CLOCK
{
    // Name, variable
    "Clock", AREA_INACTIVE | 14,
    // Actions on object
    {
        {"Examine", object_living_room_clock_examine},
    }
},
```

Dodajmy deklarację akcji w pliku actions_def.h:

```
void object_living_room_clock_examine(void);
```

Oraz implementację akcji w actions_def.c:

```
#include <stdio.h>
```

```
void object_living_room_clock_examine(void)
{
    byte text_buffer[26];
    byte h=object_value_get(OBJECT_LIVING_ROOM_CLOCK);
    sprintf(text_buffer,"The clock shows %d hour",h);
    console_print(text_buffer,true);
    if (++h > 23)
        h=0;
    object_value_set(OBJECT_LIVING_ROOM_CLOCK,h);
}
```

Jak widać po tym kodzie można używać normalnych funkcji i zmiennych języka C, ale nie jest to polecane ze względu na rozmiar generowanego kodu. Po skompilowaniu zegar odlicza przy każdym sprawdzeniu:

