

## Część 6 – inwentarz, kontenery i wybór obiektów

Większość gier przygodowych typu Point&Click umożliwia operacje na przedmiotach i przenoszenie ich pomiędzy różnymi schowkami, w tym inwentarzem gracza.

To samo umożliwia Adventure Studio.

Do obsługi inwentarza potrzebujemy mieć przypisane obrazki do obiektów i zdefiniowane kontenery. Najpierw jednak potrzebujemy przygotować grafikę inwentarza.

### Grafika inwentarza

Ponieważ aktualny Adventure Studio wyświetla inwentarz w linii poziomej, wszystkie elementy inwentarza muszą mieć ten sam kolor w liniach (ograniczenie Atari). Najprościej użyć Quantizatora do znalezienia najlepszej reprezentacji kolorystycznej całej linii, choć i tak jest trochę zabawy z preprocesem obrazków, żeby Quantizator dał dobry efekt. Tu wykorzystamy 4 obrazki (puste tło, taśma klejąca, chomik, serduszko):



Grafika została podzielona za pomocą gfx\_slicer na 4 obrazki mające 16x32 piksele i skopiowana do game\pictures jako iempty.mic, itape.mic, ihamster.mic i ilove.mic. Wspólna paleta kolorów jest skopiowana jako invent.dli.

W pliku pictures\_def.h wyliczamy kolejne obrazki:

```
PICTURE_INVENTORY_EMPTY,  
PICTURE_INVENTORY_TAPE,  
PICTURE_INVENTORY_HAMSTER,  
PICTURE_INVENTORY_LOVE,
```

i paletę:

```
PALETTE_INVENTORY,
```

W pliku pictures\_def.c definiujemy obrazki inwentarza:

```
{"iempty.mic", 16,32,PALETTE_INVENTORY,0},  
{"itape.mic", 16,32,PALETTE_INVENTORY,0},  
{"ihamster.mic",16,32,PALETTE_INVENTORY,0},  
{"ilove.mic", 16,32,PALETTE_INVENTORY,0},
```

oraz paletę:

```
{ "invent.dli", 128, 0 },
```

### Obiekty inwentarza

W tym przykładzie obiekty inwentarza nie mają swoich odpowiedników obszarowych. Nic nie stoi na przeszkodzie, aby tak było. Do “wzięcia” obiektu z ekranu trzeba wyłączyć i zamalować zabierany obszar ekranu. W przykładowej grze dołączonej do Adventure Studio jest tak z obiektem “herbs”.

Ponieważ obiekty inwentarza nie mają swoich odpowiedników obszarowych, wyliczamy je w

objects\_def.h, zaczynając od wartości AREA\_LAST\_AREA, czyli za obiektami mającymi swój obszar:

```
enum {  
    OBJECT_INVENTORY_TAPE = AREA_LAST_AREA, // first object that is not an area  
    OBJECT_INVENTORY_HAMSTER,  
    OBJECT_INVENTORY_LOVE,  
    OBJECT_LAST_OBJECT // the last object - this sets the size of the table  
};
```

W pliku objects\_def.c dodajemy definicje tych obiektów:

```
// OBJECT_INVENTORY_TAPE  
{  
    // Name, variable  
    "Duct Tape", 0,  
    // Actions on object  
    {  
        {"Use on", object_inventory_tape_use_on },  
        {TEXT_NONE, ACTION_NONE},  
    },  
},  
// OBJECT_INVENTORY_HAMSTER  
{  
    // Name, variable  
    "Hamster", 0,  
    // Actions on object  
    {  
        {"Merge with", object_inventory_hamster_merge_with },  
        {TEXT_NONE, ACTION_NONE},  
    },  
},  
// OBJECT_INVENTORY_LOVE  
{  
    // Name, variable  
    "Love", 0,  
    // Actions on object  
    {  
        {"Examine", object_inventory_love_examine },  
        {TEXT_NONE, ACTION_NONE},  
    },  
},
```

Nie definiujemy obiektu dla pustego pola inwentarza, ponieważ będzie ono odpowiadać obiektowi OBJECT\_NONE.

Oczywiście przypisane obiektom akcje musimy zaimplementować.

## **Przypisanie obiektom obrazków**

Ze względów oszczędności pamięci nie wszystkie obiekty mają przypisane im obrazki. Przypisanie to robimy w ostatniej sekcji pliku pictures\_def.h:

```
{ OBJECT_NONE, PICTURE_INVENTORY_EMPTY },  
{ OBJECT_INVENTORY_TAPE, PICTURE_INVENTORY_TAPE },  
{ OBJECT_INVENTORY_HAMSTER, PICTURE_INVENTORY_HAMSTER },  
{ OBJECT_INVENTORY_LOVE, PICTURE_INVENTORY_LOVE },
```

Ponieważ są cztery obrazki inwentarza, ustalamy też wielkość tablicy mapowania w pliku game\config\game\_cfg.h na 4:

```
#define MAX_PICTURE_OBJECT_MAPPINGS 4
```

## Zdefiniowanie kontenerów

Kontenery wyliczami w pliku `game\containers\containers_def.h`. Domyślnie znajduje się tam `CONTAINER_INVENTORY`, czyli inwentarz. Dodamy dodatkowy kontener `CABINET`.

```
enum {  
    CONTAINER_INVENTORY,  
    CONTAINER_CABINET,  
    CONTAINER_LAST_CONTAINER  
};
```

Graczowi damy do inwentarza taśmę, zaś do szafki włożymy chomika. Plik `conatiners_def.c`:

```
// CONTAINER_INVENTORY  
{  
    // Name  
    "Inventory",  
    // Objects inside  
    {  
        OBJECT_INVENTORY_TAPE,  
        OBJECT_NONE,  
    },  
},  
// CONTAINER_CABINET  
{  
    // Name  
    "Cabinet",  
    // Objects inside  
    {  
        OBJECT_INVENTORY_HAMSTER,  
        OBJECT_NONE,  
    },  
},
```

Pozostało stworzyć nowy obszar na obrazku, który będzie pełnił rolę kontenera. Będzie nim szafka pod telewizorem – `OBJECT_LIVING_ROOM_CABINET`, którą dodajemy w plikach `areas_def.h`, `objects_def.c`:

```
// OBJECT_LIVING_ROOM_CABINET  
{  
    // Name, variable  
    "Cabinet", AREA_ACTIVE,  
    // Actions on object  
    {  
        {"Get item", object_living_room_cabinet_get_item},  
        {"Put item", object_living_room_cabinet_put_item},  
    }  
},
```

Trzeba oczywiście określić wielkość obszaru w pliku `areas_def.c`:

```
// OBJECT_LIVING_ROOM_CABINET  
{ 16,85, 30,30},
```

i dodać ten obszar do definicji pokoju w pliku `rooms_def.c`

## Obsługa akcji kontenera i wybór obiektów

Kontener obsługuje się za pomocą akcji, które przypisane są do obiektu, do poleceń get i drop/put. Można to zrobić np. tak:

```
void object_living_room_cabinet_put_item(void)
{
    object_id obj;
    if (container_get_number_of_items(CONTAINER_CABINET) >= MAX_OBJECTS_IN_CONTAINER)
    {
        console_print("There is no more place in the cabinet.", true);
        return;
    }
    obj = container_object_move_with_choose(CONTAINER_INVENTORY, CONTAINER_CABINET);
    if (obj != OBJECT_NONE)
    {
        console_print("You have put a ", false);
        console_print(object_name_get(obj), false);
        console_print(" into a cabinet.", true);
    }
}

void object_living_room_cabinet_get_item(void)
{
    object_id obj;
    if (container_get_number_of_items(CONTAINER_INVENTORY) >= MAX_OBJECTS_IN_CONTAINER)
    {
        console_print("You have no place in your inventory.", true);
        return;
    }
    obj = container_object_move_with_choose(CONTAINER_CABINET, CONTAINER_INVENTORY);
    if (obj != OBJECT_NONE)
    {
        console_print("You have taken a ", false);
        console_print(object_name_get(obj), false);
        console_print(" from the cabinet.", true);
    }
}
```

W prosty sposób można połączyć dwa obiekty w jeden nowy. Zostało to dodane, ponieważ jest to czynność często wykonywana w grach przygodowych:

```
void object_inventory_tape_use_on(void)
{
    object_id obj = ui_container_choose_item(CONTAINER_INVENTORY);
    if (obj == OBJECT_INVENTORY_HAMSTER)
    {
        container_merge(CONTAINER_INVENTORY,
            OBJECT_INVENTORY_TAPE,
            OBJECT_INVENTORY_HAMSTER,
            OBJECT_INVENTORY_LOVE);
        console_print("You have have created love!", true);
    }
}
```

aby w akcji wybrać obszar z pokoju, należy użyć funkcji:

```
object_id obj = ui_object_choose(room_current_get());
```