

Part 4 – areas, objects and limits

In the third part of the tutorial we have shown the picture of the room, but the cursor pointer had no usage yet. In this part we will change it.

Defining an area

The area is determined by its position and size. Open the "game\areas" directory and to the areas_def.h file add an object that is the room:

```
////////////////////////////////////  
// IDs of areas  
////////////////////////////////////  
  
enum {  
    OBJECT_LIVING_ROOM,  
    AREA_LAST_AREA  
};
```

In areas_def.c we define the area, by setting its position (x,y) and size in pixels (width, height). In this case we want it to be the whole screen.

```
// OBJECT_LIVING_ROOM  
{ 0,0, 160,160},
```

The position on the screen can be checked by the Windows version, that continuously shows the position of the cursor.

Objects

In the Adventure Studio every area is an object, which can have some features. We have defined the area and now we have to define the corresponding object. Here we proceed in a different way than usual, and we **do not** add the object identifier to the "game\objects\objects_def.h". We do not add it, because we just gave the identifier in areas_def.h. In other words, areas are objects, but they have position on the screen set in the areas_def.c.

We have to add, however, the definition of the object. To do that we open the "game\objects\objects_def.c" file:

```
// OBJECT_LIVING_ROOM  
{  
    // Name, variable  
    "Living Room", AREA_ACTIVE,  
    // Actions on object  
    {  
        {"Examine", object_living_room_examine},  
        {"Leave", object_living_room_leave},  
    }  
},
```

In the definition of object we set its name, the default state (in case of area AREA_ACTIVE or AREA_INACTIVE), and then the actions, that can be performed on the objects (name, action procedure).

Limits

Trying to compile this code will fail, because we have not defined the actions `object_living_room_examine` and `object_living_room_leave`, but there is one more error:

error C2078: too many initializers in line `{"Leave", object_living_room_leave}`

This is caused by the limits set for the size of some array, for saving the precious memory. The limits are in “game\config\game_cfg.h” file and for the empty template from Tutorial Part 2 they are set as following:

```
#define MAX_AREAS_IN_ROOM 1
#define MAX_OBJECT_ACTIONS 1
#define MAX_OBJECTS_IN_CONTAINER 1
#define MAX_PICTURE_OBJECT_MAPPINGS 1
```

`MAX_AREAS_IN_ROOM` is the number of areas that can be created for a room. For sure we want to have a few, so we set it for to 3.

`MAX_OBJECT_ACTIONS` is the number of actions, that can be performed on an object.

Exceeding this value caused the error „*too many initializers*“. Let’s set this value to 2.

`MAX_OBJECTS_IN_CONTAINER` is the limit of the objects that can be stored in a container like player's inventory, cabinet etc. For this game we set it to 2.

`MAX_PICTURE_OBJECT_MAPPINGS` currently we keep as 1. It will be explained and set in the part of the tutorial about the inventory.

As a result we have:

```
#define MAX_AREAS_IN_ROOM 3
#define MAX_OBJECT_ACTIONS 2
#define MAX_OBJECTS_IN_CONTAINER 2
#define MAX_PICTURE_OBJECT_MAPPINGS 1
```

WARNING: When we increase the limits we have to complete the arrays with „empty values“. For example if `MAX_OBJECT_ACTIONS` were 3, the definition of the object should be completed by:

```
// Actions on object
{
    {"Examine", object_living_room_examine},
    {"Leave", object_living_room_leave},
    {TEXT_NONE, ACTION_NONE},
}
```

We have to do it, because the compiler fills the undefined elements with zeroes. In this case `TEXT_NONE` and `ACTION_NONE` are zeroes too, so it would be fine, but in other definitions, like containers, 0 is the index of the first object while `OBJECT_NONE` is defined as 0xFF. So, leaving it to the compiler can cause unexpected behaviour of the game.

Actions

We have to create action procedures for *Examine* and *Leave* commands. We do it as before – add them to actions_def.h:

```
void object_living_room_examine(void);
void object_living_room_leave(void);
```

and actions_def.c:

```
void object_living_room_examine(void)
{
    console_print("An old fashioned living room.",true);
}

void object_living_room_leave(void)
{
    console_print("The kitchen room is not defined yet!",true);
}
```

The console_print command prints the text. The second parameter *true* means, that after showing the text the game will wait for pressing „fire“ and then will clear the text console.

Assigning areas to room

We have defined the area and object actions, but we still did not assign the area to the particular room. To do it we open the „game\rooms\rooms_def.c“ file and modify the definition of our room:

```
// 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,
        OBJECT_NONE,
        OBJECT_NONE,
    }
},
```

The maximum number of areas, that can be assigned to a room, we have defined in MAX_AREAS_IN_ROOM as 3.

After compilation we can execute two actions on the area/object that is the whole screen:



In the next part we are going to manage the object properties.