

Part 3 – The first room

In this part of tutorial we will learn how to create a room and how to show a previously prepared picture.

Defining a room

The whole game definition is in the "game" directory. To create a game you don't need to modify any part of the engine (if you don't want to). In the "game" directory there are definitions of:

- actions
- areas
- containers
- objects
- pictures
- rooms

Right now we are interested in the "rooms" directory. There are two files inside: rooms_def.h and rooms_def.c

Files with .h extensions are header files in the C language. Generally speaking they have declarations that are shared among different .c files.

In our case in rooms_def.h we have to enumerate all the rooms that will be used in the game. Currently we need just a one sample room "Living room", which gets number 0 (enum is the enumerating type in the C language, which adds 1 to every next constant, by default starting from 0)

```
enum {  
    ROOM_LIVING_ROOM,  
    ROOM_LAST_ROOM  
};
```

ROOM_LAST_ROOM is the marker for the end of definitions. Size of the array containing room definitions is set by this marker. ROOM_LIVING_ROOM has the value 0, therefore ROOM_LAST_ROOM is 1.

It's time to define a room. In the rooms_def.c file we have a template Room_XXX that we change to:

```
// 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_NONE,  
    }  
},
```

room_living_room_on_draw is an action (procedure), that will be called by "on draw" event. This event is executed by automatically when entering the room, but can be also called after changes in the room, when we need to redraw the graphics covered by the inventory or when the game is loaded.

Let's ignore defining this action for now and define the picture first.

Defining a picture

Definitions of pictures and palettes are in the "game\pictures" directory in pictures_def.c and pictures_def.h. Just as in case of rooms first we have to enumerate pictures and palettes in the pictures_def.h file:

```
enum {  
    PICTURE_LIVING_ROOM,  
    PICTURE_LAST_PICTURE  
};  
  
enum {  
    PALETTE_LIVING_ROOM,  
    PALETTE_LAST_PALETTE  
};
```

Now we may add definition of the picture and it's palette to pictures_def.c.

Picture definition:

```
// PICTURE_LIVING_ROOM,  
// Filename, width, height, palette_id, data pointer  
{ "livroom.mic", 160, 128, PALETTE_LIVING_ROOM, 0 },
```

Palette definition:

```
// PALETTE_LIVING_ROOM,  
// Filename, filesize, data pointer  
{ "livroom.dli", 512, 0 },
```

The picture definition consists of file name, picture width and height, palette identifier and 0 at the end. The palette identifier is the ID of just defined palette. We rename the file from "room1.mic0,0(160,128).mic" to a shorter "livroom.mic", to fit the limits of Atari DOS. We may read the size of the image from the file name generated by the gfx_slicer tool (value in the brackets).

The palette definition consists of file name, file size and 0 at the end.

MIC and DLI files must be copied with defined names into "game\pictures" directory. From this place they are automatically copied during the compilation to the ATR image.

Showing the image after entering the room

To show the picture we must define an action. The action definitions are in the "game\actions" directory, in actions_def.h and actions_def.c files. Every action is a procedure in the following format :

```
void action_name(void);
```

In actions_def.h we add:

```
void room_living_room_on_draw(void);
```

and in the actions_def.c we implement the action:

```
void room_living_room_on_draw(void)  
{  
    picture_draw(PICTURE_LIVING_ROOM, 0, 0);  
}
```

Now we can compile, run, and see the first picture in the background of the room:



In the next part we will learn how to define areas, that can be chosen with the cursor and we will learn how to manage game objects.