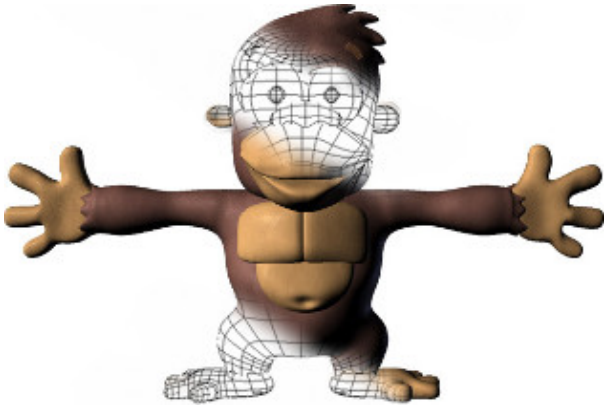
B1 - C Graphical Programming

B-MUL-100

Bootstrap My Runner

Introduction to game objects and parallax scrolling



1.0

PREAMBULE

In order to be able to do that bootstrap correctly you should have entirely finished the first initiation bootstrap and the my_hunter Bootstrap.
The exercices asked here assume that you already know :

- How to open a simple window opened.
- How to display a sprite.
- How to set a sprite texture rectangle.

This Bootstrap will help you to :

- manage your gaming entities.
- create a parallax scrolling in your new game.

Game Object

One of the most important thing when you develop a game is how you decide to manage the different entities or objects that will be used in your game

For your my_hunter, since it was your first project it probably looked like a bunch of differents structures created only to fit the coding style. And most of the time, those structures will have a lot of elements in common.

Try to create a single structure that can be put in a container of your choice (linked list, array) which will contain most / all of your game object.

Since the goal of your structure is to handle any kind basic game object it should at least contain :

- an enum with the type of the object.
- an sfSprite used for display.
- an sfTexture with the spritesheet of your object.
- an sfVector2f with the current position of your object.
- an sfIntRect with the current rectangle of your spritesheet.

Try to find the SfEventType for an enum example in the documentation !

Since all of your game object now have the same basic structure, you can create a function which follows this prototype:

```
struct game_object *create_object(const char *path_to_spritesheet,
sfVector2f pos, sfIntRect rect);
```

This function will call the appropriate CSFML functions to create your game object and return it to you. If one of your CSFML call fail your function should return `NULL`.
You can make it more complex suiting your own game object structure.

You can also create functions that return newly created sfVector2 / sfIntRect and who takes positions in parameters.

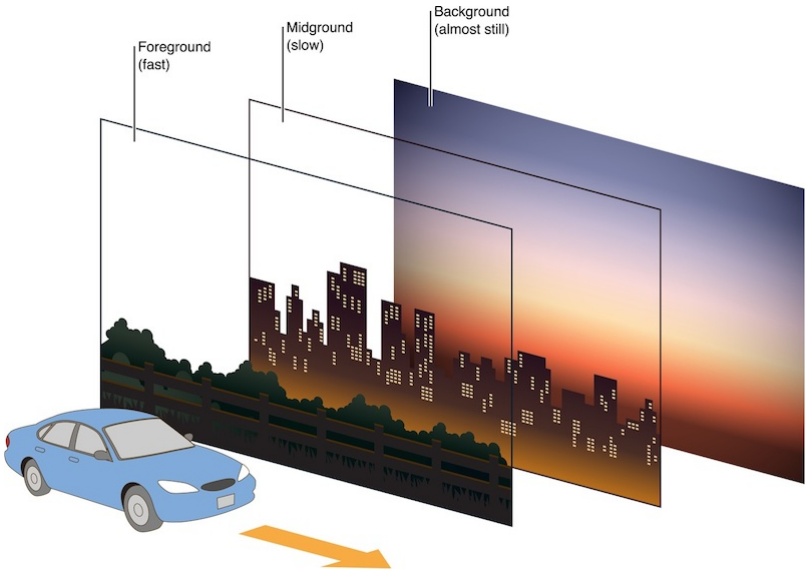
Then make a function `destroy_object` which follows this prototype:

```
void destroy_object(struct game_object *obj)
```

This function will be used when you want to clean up after you finished using an object. It should make all the appropriate calls to free your object and it's different part (like sfSprite_destroy).

The main idea is that you need to try to be as generic as possible when coding a game. Obviously some of your interraction won't be game object independant but must of them should be.

Parallax



The main goal of a runner game is to give your player an impression of speed.

To do that we use a simple concept : Parallax Scrolling. The idea is simple : having different background who moves at different speed. Trust me, it works pretty neatly ;) !

You can find sprites in the wiki !

.

We need to add something more in our game_object structure. Let's put a speed variable and a function pointer to a function that move our object.

```
struct game_object obj;

obj.type = DUCK;
obj.speed = 10f;
obj.ptr_move = &move_duck
```

The `move_duck` function will change the position of the object according to it's speed. It will also change the current sfIntRect of the object

The intressting part is that you can have different behaviours for your game objects just by giving different move functions to them !

Now create your 3 background game object, put them in your container and gave them different speeds.

Your game loop should look like this :

```
while (/* game is not over */) {
      // do some cool things

      // for an array
      obj[i].ptr_move(&(obj[i]));
      ++i;

      // for a linked list
      obj->ptr_move(obj);
      obj = obj->next;
}
```

See how cool it is ? No need to manage what's inside your game object. Your move function will do it for you :) ! You can obviously apply the same mechanics for different functions (display, physics, ...) but think about it before :) !

You should have a nice parallax scrolling ! Well ... almost. Now it's your job to make it work ... infinitely :) !

Let your parallax demo run for a while. What's happening ?
Do you need more informations in your move functions ?