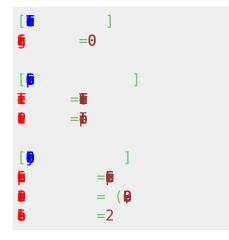# Part 20 - Text and Game Over

For our last hurrah, we'll add a score object in the top left corner of the screen and increase it for each monster shot and when the star is collected.

First the score text, graphic and object:

```
[Score]
Text        = 0

[ScoreGraphic]
Text        = Score
Pivot       = top

[ScoreObject]
Graphic     = ScoreGraphic
Position    = (ScoreObject)
Scale       = 2
```

Interesting to note the special Text definition. ScoreGraphic uses this instead of a texture from file. And the ScoreObject uses the graphic section as others before.

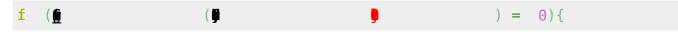Now, in code, add a score variable and a scoreObject at the top of the file:

```
orxOBJECT       *score;
orxOBJECT       *scoreObject;
orxOBJECT       *scoreObject;
orxU32          = 0;
```

Next, create a score object and assign it to the scoreObject variable in the Init() function:

```
score     = orxObject_CreateFromConfig(ScoreObject);
score     = (orxOBJECT *)orxObject_GetUserData(score);
scoreObject = orxObject_CreateFromConfig(ScoreObject);
```

Add a little function to increase the score and update the ScoreObject:

```
void UpdateScore(int score){
    score           += score;

    orxCHAR         [6];
    orxString       (orxString (orxString), score);

    orxObject       (scoreObject);
}
```

Add 250 points whenever a monster is hit with a bullet. Add UpdateScore() function calls to:

```
if (orxObject (orxString (orxString) == 0){
```

```
                              (                  );
                              (            0);
                              (             0);
               (    );
}

if (                  (                    ) =  0){
                    (                    );
                    (       0);
                    (          0);
               (    );
}
```

And of course, 1000 points bonus if the star is reached. Add the following code:

```
if (              (              ) =  0){
               (     0);
          (0  );
}

if (              (              ) =  0){
          (     0);
          (0  );
}
```

Compile and run it. Shoot some monsters and collect the star and observe the score increasing.

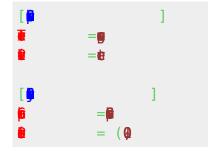That takes care of having an active score object.

A final thing for the game will be to add a game over panel when the hero dies. For this, we'll have an object that gets created using a timeline track after a two second delay. First, a game over asset:

Save this into the   folder as   .
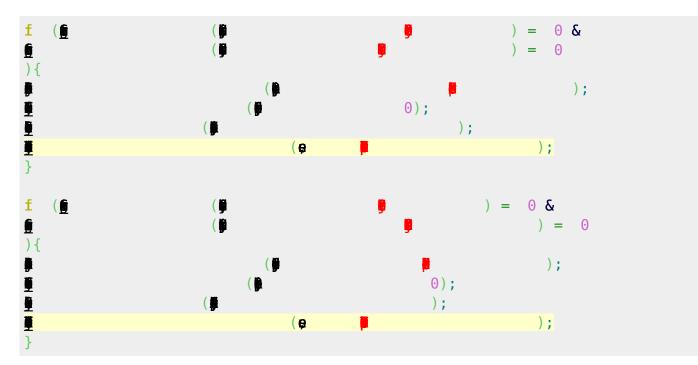
Create a simple gameover graphic and object:

```
[            ]
=
=

[            ]
=
= (0          )
```

Create a timeline track with a single command to create the   :

```
[                         ]
2 =
```

Then finally, change the physics handler code, so that when the hero is destroyed, create the timeline track. After a two second delay, the   will be created on screen:

```
f  (                 (              )  =   0  &
                     (              )  =   0
){
                        (                    );
                (            0);
                   (                 );
                        (                  );
}

f  (                 (              )  =   0  &
                     (              )  =   0
){
                        (                    );
                (            0);
                   (               );
                        (                  );
}
```

Because we never had a reference to the   before in order to add the PopUpGameOverTrack, we'll need to make a variable reference to it at the top of the code:
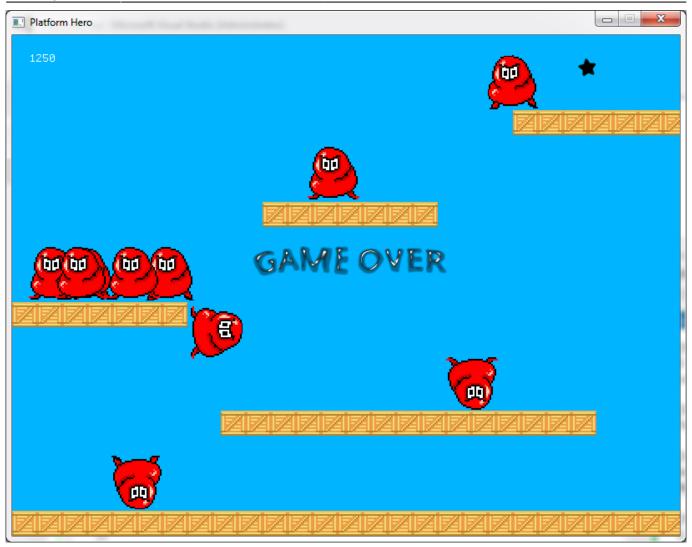
```
             *        ;
             *          ;
             *       ;
             *           ;
                   = 0;
```

And then set the   variable on the orxObject_CreateFromConfig("Scene") in the
function:

```
            =                           (               );
                 (             );
      =                     (         );
```

Compile and run. Kill your hero and wait two seconds. The game over panel will appear:

Well friends, that's game over. If you reached the end here, great job! You've learned many of the major features of Orx.

If you need more help, go to the tutorials section where many of these concepts are covered in greater detail.

If you need quick examples, you can visit the examples section and search by subject.

If you get stuck, please post over on our Discord. Hope to see you there.

- Part 1 – Downloading Orx
- Part 2 – How Orx works
- Part 3 – Setting up a new game project
- Part 4 – A tour of an Orx project
- Part 5 – Viewport and the camera
- Part 6 – Objects
- Part 7 – Spritesheets and Animation
- Part 8 – Platforms and Texture Repeating
- Part 9 – Physics
- Part 10 – Input Controls