Part 19 - Survival

Keeping alive is key in a platformer. The monsters need to be a threat to our hero. If a monster touches our hero, he should explode in a shower of bits. That's game over.

For the explosion for our hero, let's use this object from the `orx/tutorial/data/object` folder in the Orx project:



Might seem an odd choice, but if 50 of these spawned over the hero in various stages of rotation and flew in several directions like sparks, it will be fairly effective. Start with the object and the graphic:

```
[SparkGraphic]
Texture = +.png
Pivot   = center

[SparkObject]
Graphic  = SparkGraphic
Speed    = (-350, -350, 0) ~ (350, -850, 0)
Color    = (255, 0, 0) ~ (255, 255, 255)
Rotation = 0 ~ 90
LifeTime = 2 ~ 4
Scale    = 0.5 ~ 1.0
Body     = SparkBody
```

Each `SparkObject` will fly up and out in a random direction, random colours, sizes, and a random lifetime. 50 at a time should look good. Next, make a body for it so that it will fall back to earth:

```
[SparkBody]
Dynamic  = true
PartList = SparkBodyPart

[SparkBodyPart]
Type  = box
Solid = false
```

A pretty simple body and part - affected by gravity but not set to collide with anything.



Note that giving each particle a body is an easy way to make objects fall to the ground, but it is also heavy and in-efficient. While we won't do it in this guide, an FX can also be a cheap way to make object fall down. See this tutorial if you would like to learn how: Creating Electrical Sparks

Finally, we'll make an empty hero exploder with a spawner that spawns out 50 `SparkObject`'s:

```
[HeroExploder]
Spawner = HeroSpawner

[HeroSpawner]
Object          = SparkObject
WaveSize        = 50
WaveDelay       = 0.1
TotalObject     = 50
```

Before we can deal with the collision between monster and hero, we need to ensure the flags of the hero knows about monsters:

```
[HeroBodyPart]
Type        = box
Solid       = true
SelfFlags   = hero
CheckMask   = platforms # star # monster
```

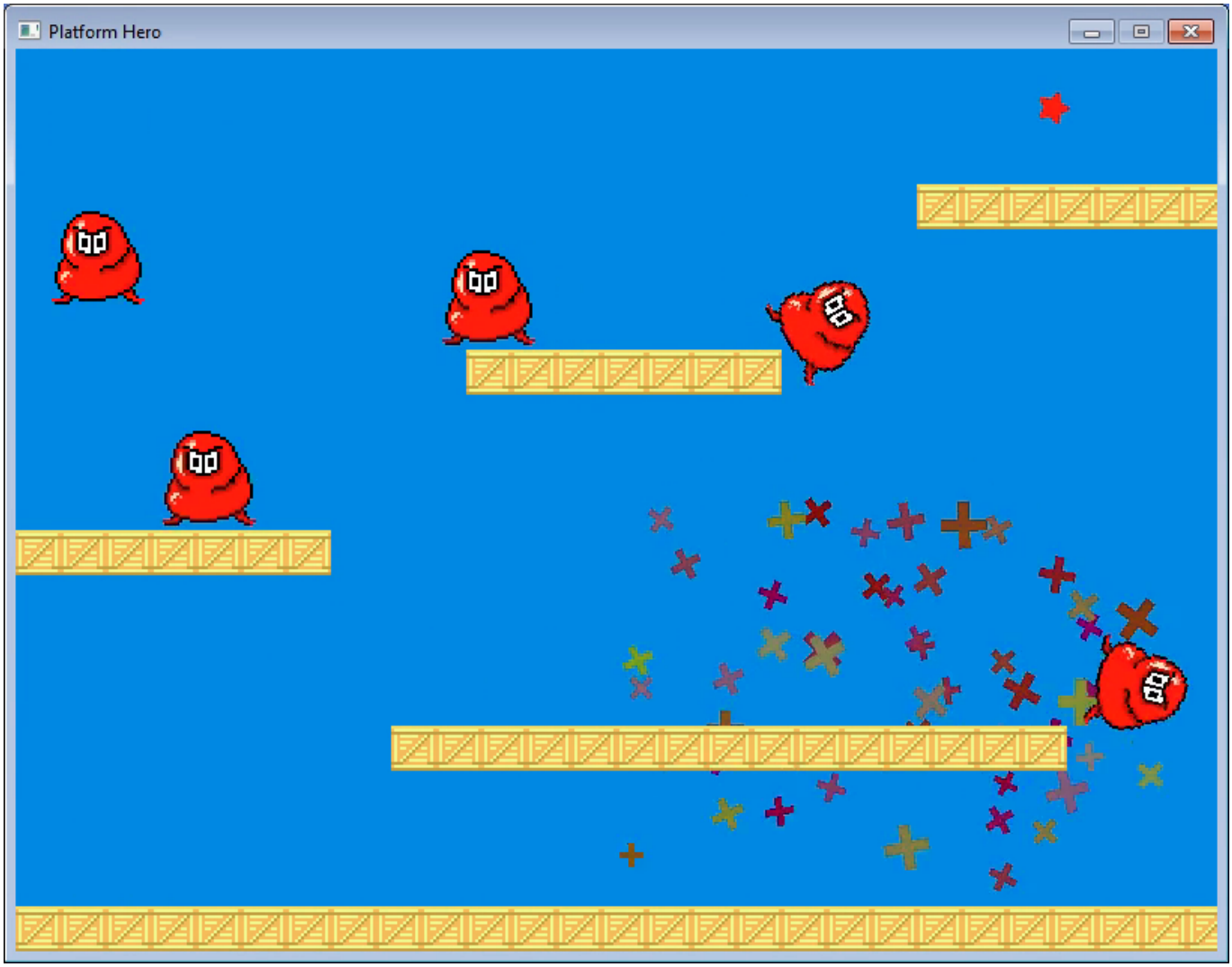
Now to handle it in code, by adding the following at the bottom of the `PhysicsEventHandler` function:

```
if (orxString_Compare(recipientObjectName, "HeroObject") == 0 &&
    orxString_Compare(senderObjectName, "MonsterObject") == 0
){
    CreateExplosionAtObject(pstRecipientObject, "HeroExploder");
    orxObject_SetLifeTime(pstSenderObject, 0);
    orxObject_Enable(pstRecipientObject, orxFALSE);
}

if (orxString_Compare(senderObjectName, "HeroObject") == 0 &&
    orxString_Compare(recipientObjectName, "MonsterObject") == 0
){
    CreateExplosionAtObject(pstSenderObject, "HeroExploder");
    orxObject_SetLifeTime(pstRecipientObject, 0);
    orxObject_Enable(pstSenderObject, orxFALSE);
}
```

So if the hero and the monster touch, disable the hero and place a `HeroExploder` on top of him.

Compile and run it:



So there you go. The major elements of the game are now complete. Well done!

In our next and last chapter, we'll do a few housekeeping changes to flesh the game out.

Next: Part 20 – Text and Game Over.

- 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
- Part 11 – Running and Standing
- Part 12 – Changing Direction
- Part 13 – Getting our hero to shoot

- Part 14 – FX
- Part 15 – Collision Events.
- Part 16 – Jelly Monsters
- Part 17 – Timeline Tracks
- Part 18 – Exploding Monsters
- Part 19 – The Hero's survival.
- Part 20 – Text and Game Over

From:
https://orx-project.org/wiki/ - **Orx Learning**

Permanent link:
https://orx-project.org/wiki/en/guides/beginners/survival?rev=1530285796

Last update: **2025/09/30 17:26 (14 months ago)**