Part 9 – Physics

For our game we will need to add physics to keep our hero firmly on the ground.

The first thing to do is to enable physics in the config:

```
[Physics]
AllowSleep          = false
Gravity             = (0.0, 2400.0, 0.0)
```

This will set the strength and direction of the gravity and stops the physics being applied when objects settle. Our gravity will be pretty strong at 2400.

In order for physics to act on an object, the object needs to have a body. Let's add one to our hero:

```
[HeroObject]
Graphic      = HeroGraphic
Position     = (-350, 100, 0)
Scale        = 2
AnimationSet = HeroAnimationSet
Body         = HeroBody
```

Now to define the HeroBody section:

```
[HeroBody]
Dynamic  = true
PartList = HeroBodyPart
```

Dynamic means that the object is affected by physics and gravity. Each Body must have at least one body part. Define that next:

```
[HeroBodyPart]
Type  = box
Solid = true
```

The type is box and that means the collision area is a box shape surrounding the object. The part is also set to solid. That means that the object will not be able to pass through other solid body parts.

Run that and you'll see our hero fall right through the floor. So that's good. The physics body is working. But we need to get the platforms solid too.

```
[PlatformObject]
Graphic  = PlatformGraphic
Position = (-400, 270, 0)
Scale    = (54, 2, 0)
Repeat   = (27, 1, 0)
Body     = PlatformBody
```

Now the body and body part for a platform object:

```
[PlatformBody]
Dynamic  = false
PartList = PlatformBodyPart

[PlatformBodyPart]
Type  = box
Solid = true
```

Notice the slight difference from the HeroBody? The PlatformBody is not dynamic, so it won't be affected by gravity. But the PlatformBodyPart is set to solid. This will allow the hero to collide with the platform without it falling away.

But the two won't affect each other until you set collision flags.

First, set the flags on the HeroBodyPart:

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

The SelfFlags property is a string that describes the object and the CheckMask is a mask list of other object self flags it can collide with.

Now set the PlatformBodyPart flags:

```
[PlatformBodyPart]
Type        = box
Solid       = true
SelfFlags   = platforms
CheckMask   = hero
```

So both objects are set to collide with each other.

Run the game and our hero will land and stay on the floor.

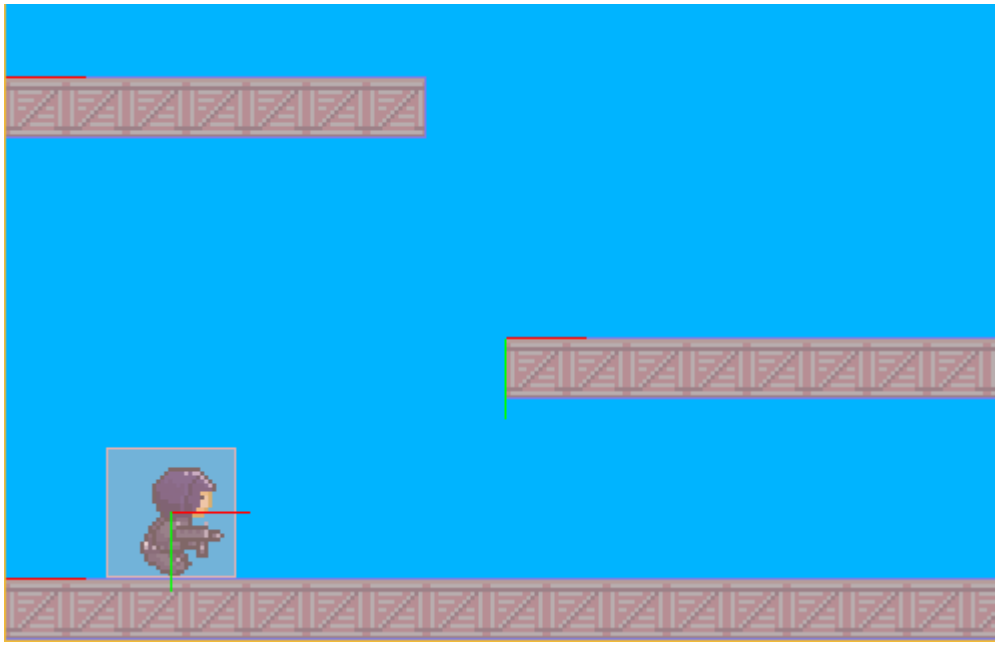
Our guy lands and stays on the floor. Good.

You can turn on physics debug mode if you want to see the bounding boxes around the objects. Turn on the physics debug with:

```
[Physics]
AllowSleep          = false
Gravity             = (0.0, 2400.0, 0.0)
ShowDebug           = true
```

You can see the boxes that surround the hero and platforms. That is because they have bodies.

Run that up and you should get the following:



Much better. Don't forget to turn off the physics debugging before continuing.

Next, Part 10 - Input Controls.

- 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://www.orx-project.org/wiki/ - **Orx Learning**

Permanent link:
https://www.orx-project.org/wiki/en/guides/beginners/physics?rev=1530168502

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