Part 10 - Input Controls

Now to add some keyboard control so that our hero is able to jump and run around.

First we'll need to define some keys. You'll notice that one key (quit) is already defined in the `[Input]` section, which is used as the default input set.

Let's add a few more:

```
[Input]
KEY_ESCAPE   = Quit
KEY_LEFT     = GoLeft
KEY_RIGHT    = GoRight
KEY_LCTRL    = Shoot
KEY_LSHIFT   = Jump
```

This assigns labels to keys. These labels can be accessed in the code.

You can add code the `Update()` function to detect these keys. The `Update()` function is tied to the Orx core clock so the keys can be checked on every frame:

```
if (orxInput_IsActive("GoLeft"))
{
}

if (orxInput_IsActive("GoRight"))
{
}
```

In order to affect the hero object, it will need to be assigned to a variable so that it can be worked with. Change the HeroObject creation line in Init() to be:

```
hero = orxObject_CreateFromConfig("HeroObject");
```

And declare the variable at the top:

```
#include "orx.h"
#include "orxExtensions.h"

orxOBJECT *hero;
```

Now it is possible to affect the object in code. We'll add a vector direction as a speed to the object based on the key pressed:

```
orxVECTOR leftSpeed = { -20, 0, 0 };
orxVECTOR rightSpeed = { 20, 0, 0 };

if (orxInput_IsActive("GoLeft"))
{
```

```
    orxObject_ApplyImpulse(hero, &leftSpeed, orxNULL);
}

if (orxInput_IsActive("GoRight"))
{
    orxObject_ApplyImpulse(hero, &rightSpeed, orxNULL);
}
```

Compile and run. Our hero will run left and right with the cursor keys. Sweet! Except he runs off really quick and doesn't stop.

He needs some damping on his body to slow him down when a speed is not being applied:

```
[HeroBody]
Dynamic           = true
PartList          = HeroBodyPart
LinearDamping     = 5
```

Run that and our hero will decelerate to a quick stop when no key is pressed.

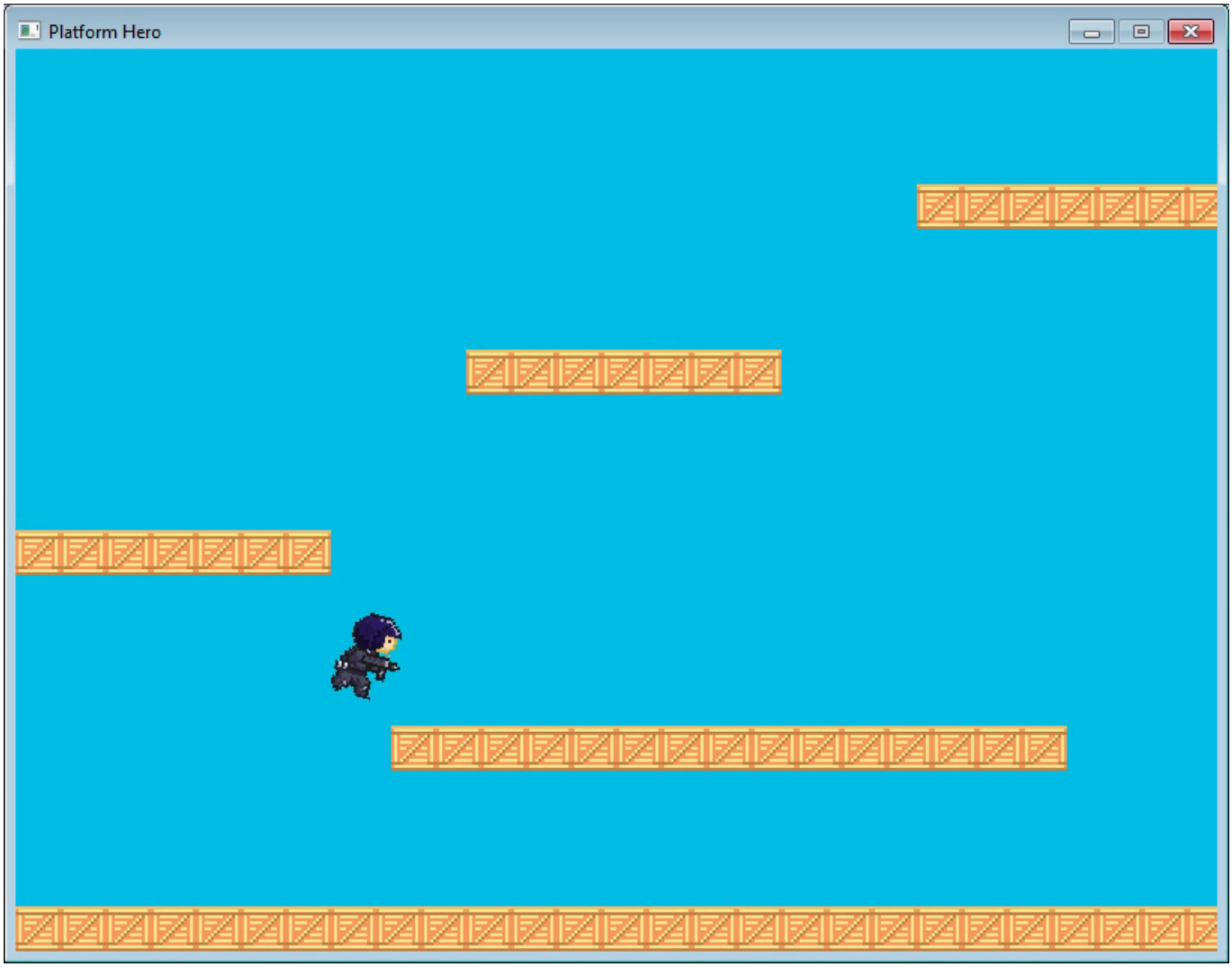
Next, we can add a jump to the `update()` function:

```
orxVECTOR jumpSpeed = { 0, -600, 0 };

if (orxInput_HasBeenActivated("Jump"))
{
    orxObject_ApplyImpulse(hero, &jumpSpeed, orxNULL);
}
```

This code is like the previous key input code but this one only applies impulse just the once when the key is pressed - not if it's held down.

Compile and run. Your hero can now jump around:



Cool he can jump around using the shift key. But our hero turns over when he hits a platform edge. We can fix that by fixing his rotation:

```
[HeroBody]
Dynamic             = true
PartList            = HeroBodyPart
LinearDamping       = 5
FixedRotation       = true
```

Run it again. Much better!

Next: Part 11 – Running and Standing.

- 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/input_controls

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