Part 15 - Collision Events

In order for the game to be won by the hero by collecting the star, we need to create a Physics Event Handler in code to check for, and to process a collision between those two objects.

We can declare one at the bottom of the `Init()` function:

```
orxViewport_CreateFromConfig("MainViewport");

orxEvent_AddHandler(orxEVENT_TYPE_PHYSICS, PhysicsEventHandler);
```

What this means is, every time there is a physics event like a collision, execute the `PhysicsEventHandler` function. We don't have one of these, so you can add one:

```
orxSTATUS orxFASTCALL PhysicsEventHandler(const orxEVENT *_pstEvent)
{

    return orxSTATUS_SUCCESS;
}
```

The type of event we are interested in for collisions is the `orxPHYSICS_EVENT_CONTACT_ADD`.

Add this code inside the new function:

```
orxSTATUS orxFASTCALL PhysicsEventHandler(const orxEVENT *_pstEvent)
{
  if (_pstEvent->eID == orxPHYSICS_EVENT_CONTACT_ADD) {
    orxOBJECT *pstRecipientObject, *pstSenderObject;

    pstSenderObject = orxOBJECT(_pstEvent->hSender);
    pstRecipientObject = orxOBJECT(_pstEvent->hRecipient);

    orxSTRING senderObjectName =
(orxSTRING)orxObject_GetName(pstSenderObject);
    orxSTRING recipientObjectName =
(orxSTRING)orxObject_GetName(pstRecipientObject);

    if (orxString_Compare(senderObjectName, "StarObject") == 0){
        //do something
    }

    if (orxString_Compare(recipientObjectName, "StarObject") == 0){
        //do something
    }
  }
  return orxSTATUS_SUCCESS;
}
```

Whenever there is contact between two objects, orxPHYSICS_EVENT_CONTACT_ADD is the physics

event ID that will trigger. Both the sender object and the recipient object are available to us in the event package.

Next, we can check if the name of either object is “StarObject”. And if so, the game will be won.

Replace the “do something” lines with a command that will instantly remove the star from the game:

```
if (orxString_Compare(senderObjectName, "StarObject") == 0){
    orxObject_SetLifeTime(pstSenderObject, 0);
}

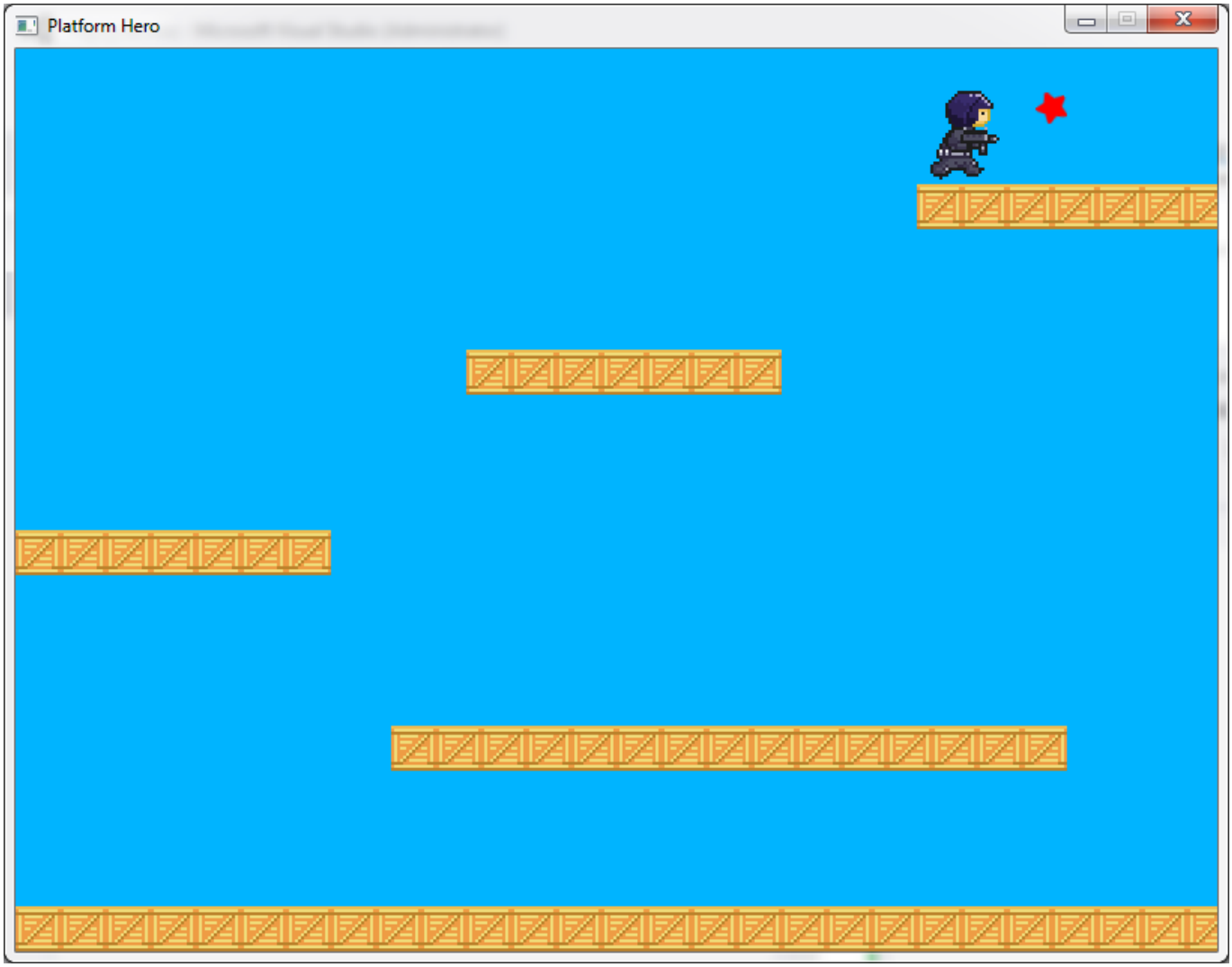
if (orxString_Compare(recipientObjectName, "StarObject") == 0){
    orxObject_SetLifeTime(pstRecipientObject, 0);
}
```

The one last thing to add to ensure the collision between the star and the hero works, is to add “star” to the HeroBody check mask:

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

Checkmasks and flags must work both ways, that is, it must be defined on both object bodies.

Compile and run. Make your way up to the top platform and the star should disappear when the hero touches it:



Next: Part 16 – Jelly Monsters.

- 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/collision_events

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