Part 18 - Exploding Monsters

Time to have the monsters explode into bits of jelly when we shoot them. Another new asset:

Right click and save this image into the `data/texture` folder of our project as “jelly.png”.

Every monster will explode into several of these jelly objects when shot.

Create graphics for the jelly frames:

```
[JellyGraphic]
Texture        = jelly.png
TextureOrigin  = (0, 0, 0)
TextureSize    = (32, 32, 0)
Pivot      = center
```

Then an individual `JellyObject`:

```
[JellyObject]
Graphic      = JellyGraphic
AnimationSet = JellyAnimationSet
Speed        = (-50, -50, 0) ~ (50, -50, 0)
Position     = (0, -300, 0)
Body         = JellyBody
```

Next to make the JellyAnimationSet:

```
[JellyAnimationSet]
Texture            = jelly.png
FrameSize      = (32, 32, 0)
JellyWobbleAnim        = -1
StartAnim      = JellyWobbleAnim
JellyWobbleAnim->    = JellyWobbleAnim
Pivot          = center

[JellyWobbleAnim]
KeyDuration = 0.08
```

The jelly needs a body. This is so the bits will bounce nicely over the platforms:

```
[JellyBody]
Dynamic  = true
PartList = JellyBodyPart

[JellyBodyPart]
Type        = sphere
```

```
Radius      = 10
Solid       = true
SelfFlags   = jelly
CheckMask   = platforms
```

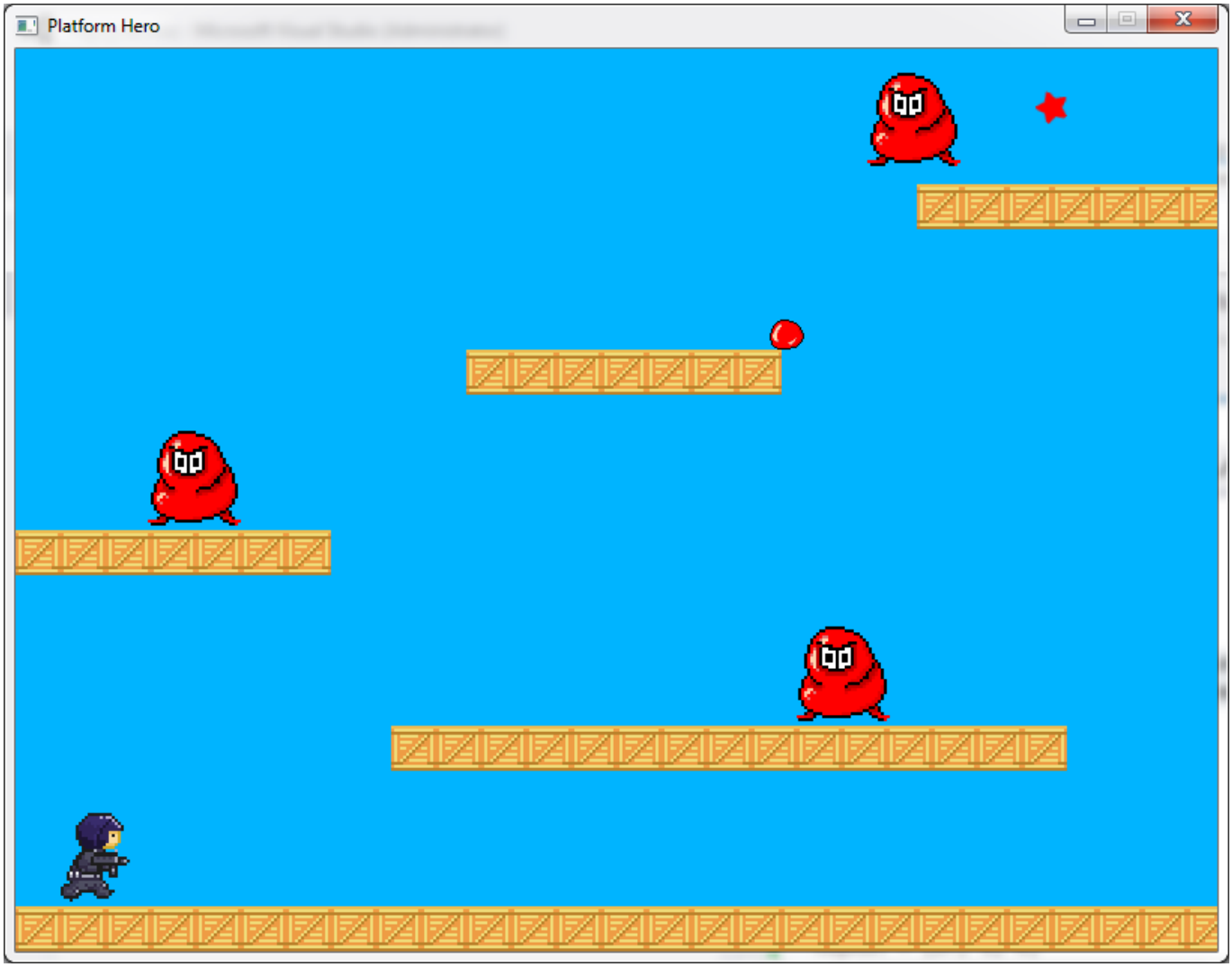
Then let the platforms know about the jelly, by adding it the platform's mask:

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

Test it by adding a JellyObject to the Scene section:

```
[Scene]
ChildList = PlatformObject # MiddlePlatformObject #
TopLeftPlatformObject # TopPlatformObject #
TopRightPlatformObject #
StarObject # JellyObject
TrackList = MonsterMakerTrack
```

Run it and a jelly with drop onto the platform:



That's not bad, but really... the animation frame setup isn't quite what we're after. In our `JellyAnimationSet` we're just letting Orx automatically take the three frames in the sprite sheet. We really need four frames:

- Frame 1
- Frame 2
- Frame 1
- Frame 3

That is how the animation should be. Thankfully Orx will let us specify the third and fourth frames for an animation manually. Add the following config to set this up:

```
[JellyWobbleAnim3]
TextureOrigin = (0, 0, 0)

[JellyWobbleAnim4]
TextureOrigin = (64, 0, 0)
```

Great. By using the name of the animation (`JellyWobbleAnim`) + the frame number, now frames 3 and 4 are overridden in the sheet while frames 1 and 2 are picked up automatically by Orx.

The next step is to create an explosion object. This will be an empty object that contains a spawner. The spawner will shoot out five jellies. These explosion objects can be placed on any monster for

dramatic effect:

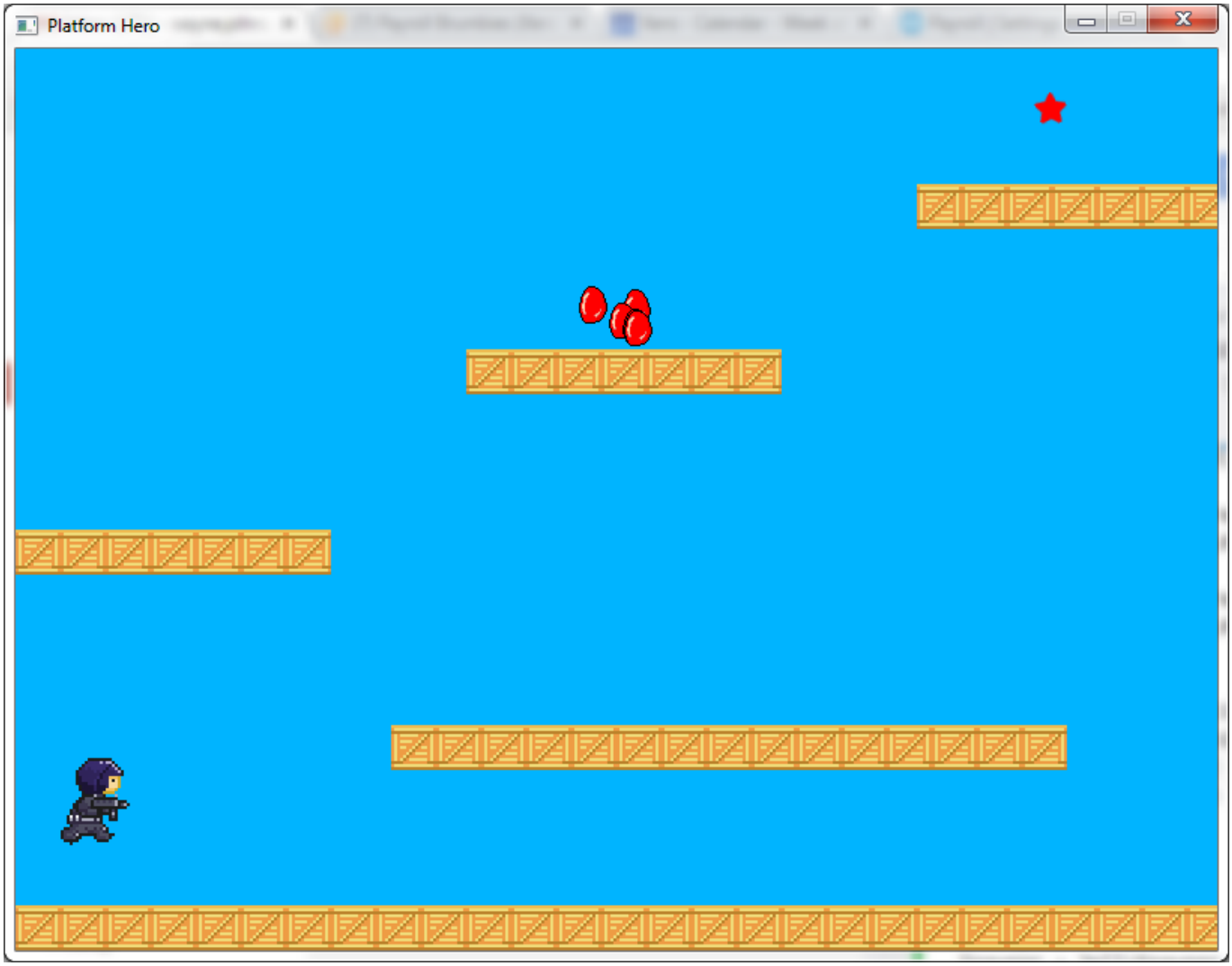
```
[JellyExploder]
Spawner          = JellySpawner

[JellySpawner]
Object           = JellyObject
WaveSize         = 5
WaveDelay        = 0.1
TotalObject      = 5
```

To test one of the these, remove the JellyObject from the Scene section and add the JellyExploder instead:

```
[Scene]
ChildList = PlatformObject # MiddlePlatformObject #
TopLeftPlatformObject # TopPlatformObject #
TopRightPlatformObject #
StarObject # JellyExploder
TrackList = MonsterMakerTrack
```

Run the game to test the exploder:



Great, we see a bunch of jelly blobs appear. Remove the JellyExploder from the Scene selector:

```
[Scene]
ChildList = PlatformObject # MiddlePlatformObject #
TopLeftPlatformObject # TopPlatformObject #
TopRightPlatformObject #
StarObject
TrackList = MonsterMakerTrack
```

Also remove the `Position` parameter from the `JellyObject` because these will soon be dynamically placed. Also change the Speed parameter to be more dramatic:

```
[JellyObject]
Graphic      = JellyGraphic
AnimationSet = JellyAnimationSet
Speed        = (-50, -250, 0) ~ (50, -450, 0)
Body         = JellyBody
```

Next, we need to ensure that the BulletObject has a body and that it can collide with a monster:

```
[BulletObject]
Graphic  = BulletGraphic
LifeTime = 1.0
Scale    = 0.25
Body     = BulletBody


[BulletBody]
Dynamic  = false
PartList = BulletBodyPart

[BulletBodyPart]
Type       = box
Solid      = false
SelfFlags  = bullet
CheckMask  = monster
```

The bullets are not affected by gravity (not Dynamic) nor do they bounce off other objects (not Solid).

In order to make a monster explode, we can make a function in the code which will create a exploder object on top off a monster object, and then destroy the monster itself:

```
void CreateExplosionAtObject(orxOBJECT *object, orxSTRING
exploderObjectName)
{
    if (object == orxNULL)
        return;

    orxVECTOR objectVector;
    orxObject_GetWorldPosition(object, &objectVector);
    objectVector.fZ = 0.0;
```

```
    orxOBJECT *explosion = orxObject_CreateFromConfig(exploderObjectName);

    orxObject_SetPosition(explosion, &objectVector);
}
```

We are passing in a named exploderObjectName so that we can re-use this function later for other exploders we'll make.

Now to use it. In the physics event, add a check for collision between a bullet and a monster and process it:

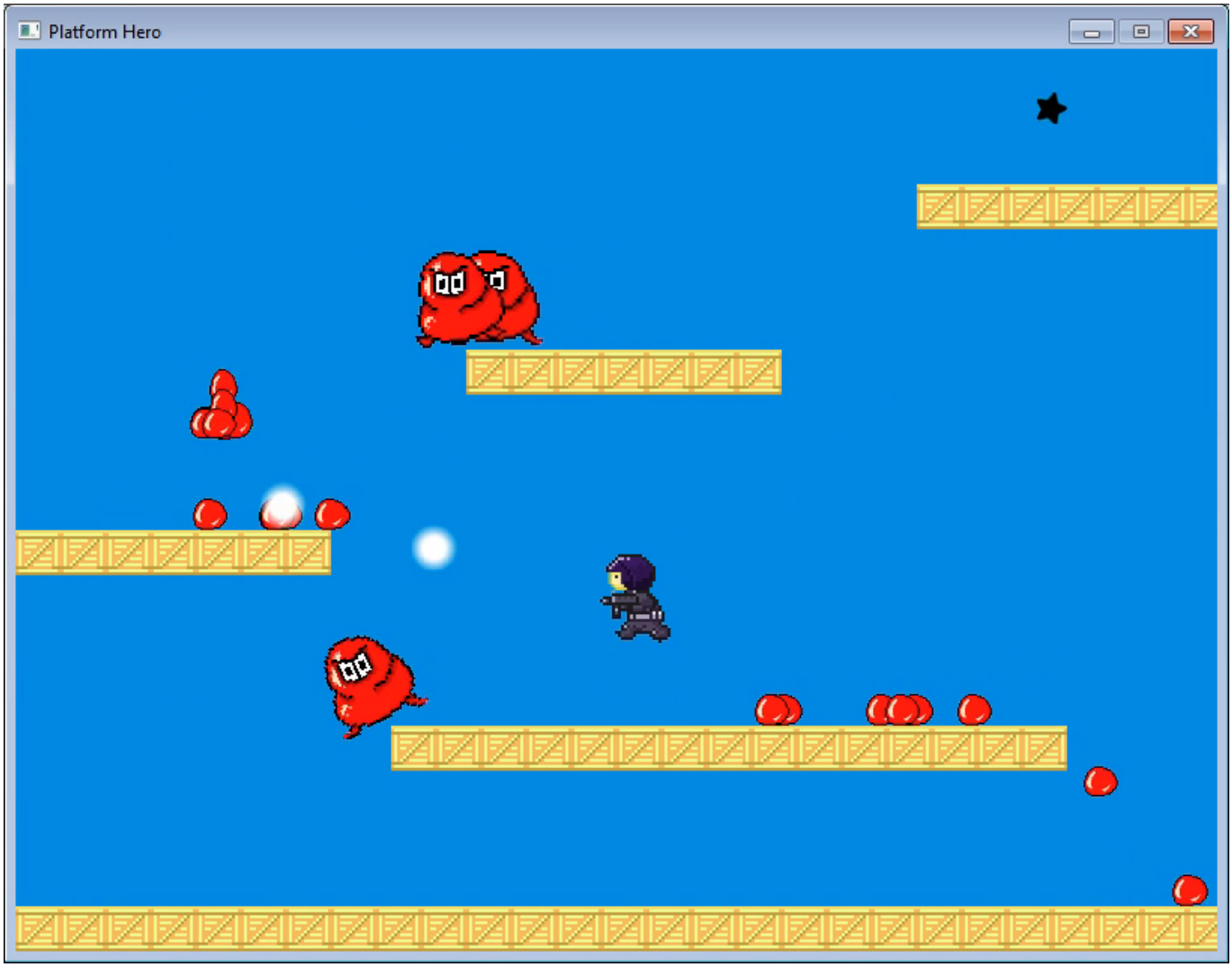
```
if (orxString_Compare(recipientObjectName, "StarObject") == 0){
    orxObject_SetLifeTime(pstRecipientObject, 0);
}

if (orxString_Compare(senderObjectName, "BulletObject") == 0){
    CreateExplosionAtObject(pstRecipientObject, "JellyExploder");
    orxObject_SetLifeTime(pstSenderObject, 0);
    orxObject_SetLifeTime(pstRecipientObject, 0);
}

if (orxString_Compare(recipientObjectName, "BulletObject") == 0){
    CreateExplosionAtObject(pstSenderObject, "JellyExploder");
    orxObject_SetLifeTime(pstSenderObject, 0);
    orxObject_SetLifeTime(pstRecipientObject, 0);
}
```

So in both cases, create the `JellyExplosion` on top of a monster, then destroy both the monster and the bullet that hit it.

Compile and run. You'll get some crazy jelly action all over the screen:



The jelly blobs never disappear. Fix that by giving it a specific lifetime:

```
[JellyObject]
Graphic      = JellyGraphic
AnimationSet = JellyAnimationSet
Speed        = (-50, -250, 0) ~ (50, -450, 0)
Body         = JellyBody
LifeTime     = 5
```

Next: Part 19 - The Hero's survival.

- 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