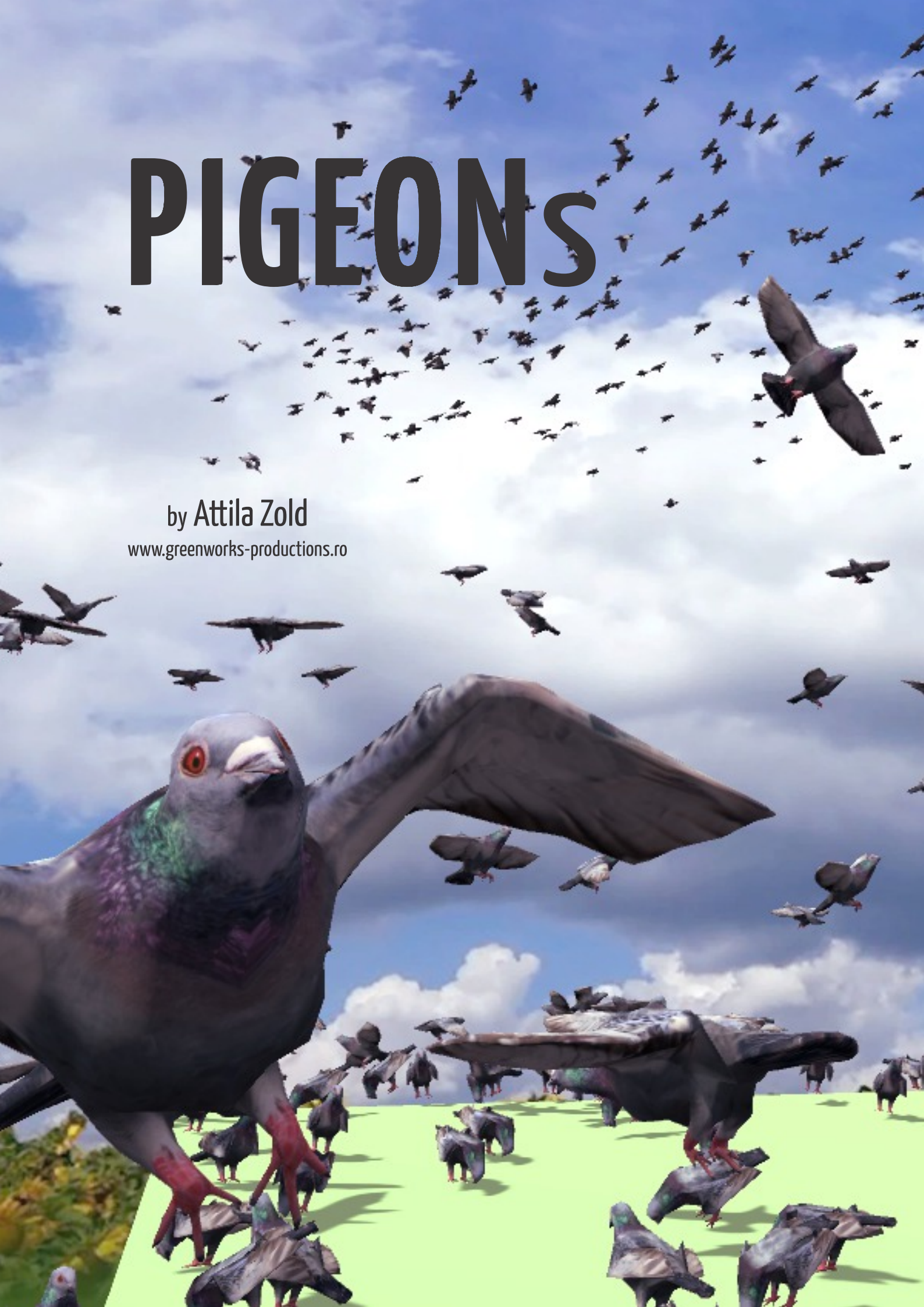
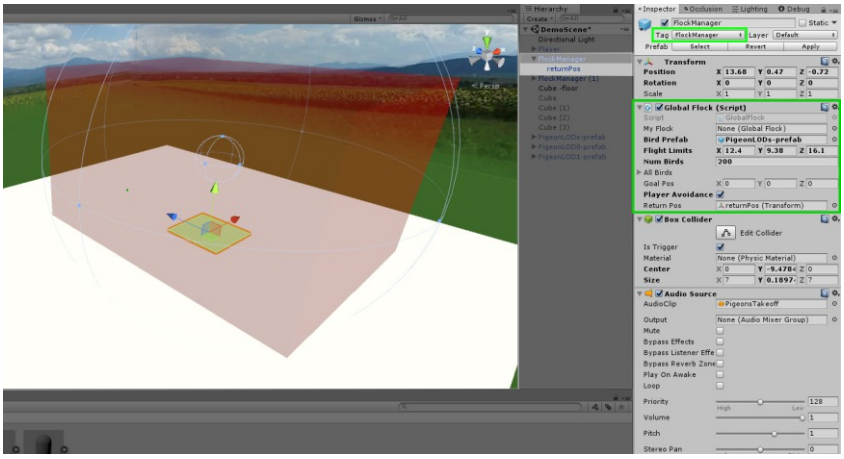
PIGEONs

by Attila Zold
www.greenworks-productions.ro



Flock Manager explained

You will find the "**FlockManager.prefab**" prefab in the Prefabs folder. It is already setup, but you can modify it, or create your own.



This prefab needs to have it's **Tag** set to "FlockManager".
Please add a new tag, name it **FlockManager**, and assign it.

The "**GlobalFlock**" **script** is attached to this prefab.

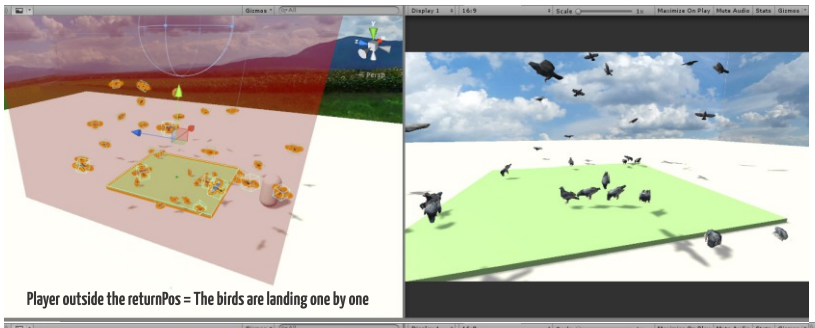
1. You need to assign one of the pigeon prefabs (or your own custom prefab)
into the **Bird Prefab** field.

2. Set the **Flight Limits** (red box) X, Y and Z values to determine the bounds inside which this flock will fly in.

3. **Num Birds** field sets the amount of birds to be instantiated.

4. **Player Avoidance**: leave this unchecked if you simply want your flock of birds to fly around without landing.

This prefab has a green colored Mesh Renderer on the "**returnPos**" child object, as an indicator for easy setup. You can disable the Mesh Renderer later.



Player outside the returnPos = The birds are landing one by one

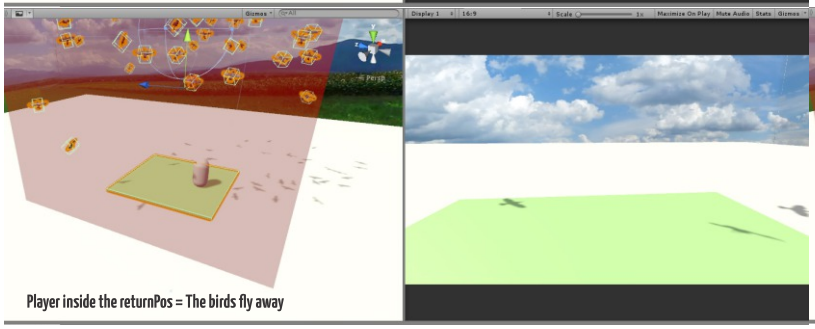
Player Avoidance:

1. Check this bool if you want the flock to have a landing zone, from which the player can scare them off of. Player needs to be **tagged as "Player"**

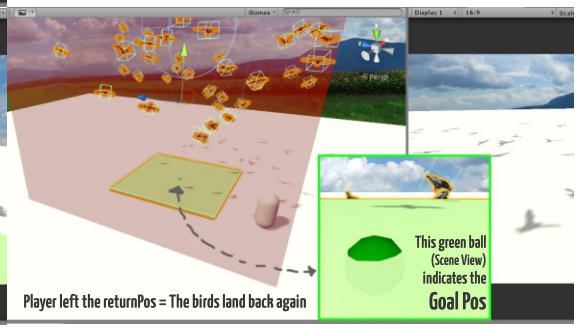
2. If checked, you must assign a transform in the **Return Pos** field, which will be the "landing zone" for this flock of birds.

3. The **Box Collider** component, which is set to **Is Trigger**, defines the bounds of the "landing zone".

4. You need to set it's bounds to match the position of your **Return Pos**.



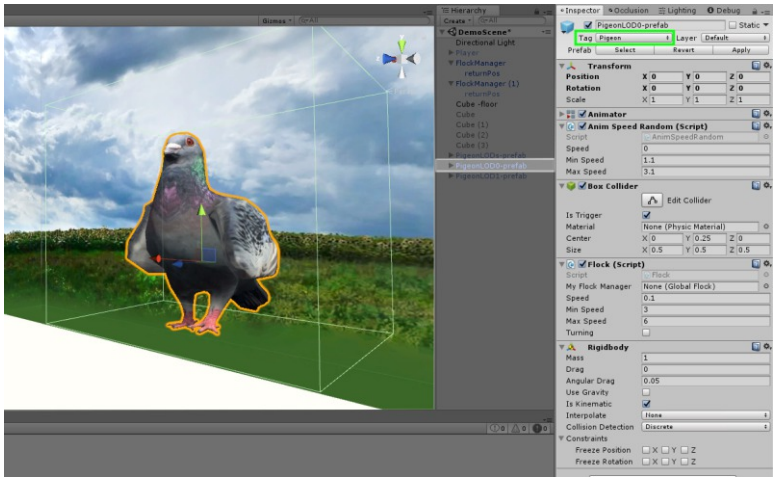
Player inside the returnPos = The birds fly away



Player left the returnPos = The birds land back again

Bird Prefab explained

Each bird prefab you setup needs to have the following components: "**AnimSpeedRandom**" script, "**Box Collider**" set to **Is Trigger**, "**Flock**" script and "**Rigidbody**" set to **NOT Use Gravity** and **Is Kinematic**. They also need to be **tagged as "Pigeon"** to be detected by the "landing zone".



A. The "**AnimSpeedRandom**" **script** sets this object's Animator's speed, based on the interval you set in the **Min Speed** & **Max Speed** fields, meaning every bird will animate at a slightly different speed (instead of all at the same time, which would be weird).

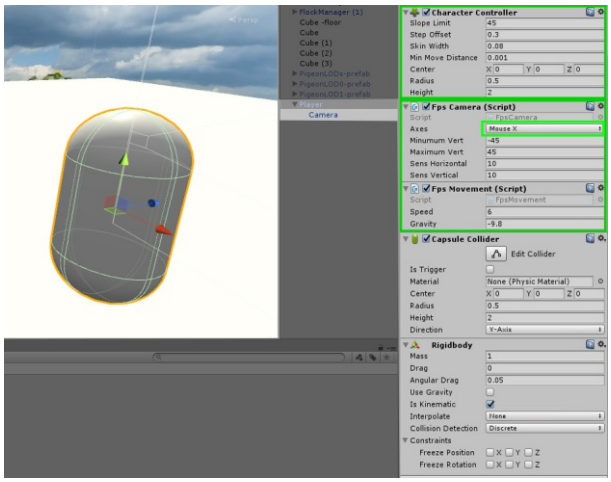
B. The **Box Collider (is trigger)** component detects the "landing zone", which you setup in the **FlockManager**'s Box Collider bounds. When the bird collides with the "landing zone", it deactivates it's Flock script, and switches anim to landed.
This gets canceled when the player enters the "landing zone".

C. The "**Flock**" **script** is the main component which makes your bird fly in a flock. Whenever this script is enabled, the bird behaves as it was flocking.
You can set a **general speed** parameter in the **Speed** field, but you can also override this by setting up an **interval** in the **Min Speed** & **Max Speed** fields.

D. A **Rigidbody** component must be present, with **Use Gravity** **unchecked** and **Is Kinematic checked**, for correct collider trigger detections.

Fps Camera/Movement setup

This asset pack comes with a simple FPS movement script, which allows you to setup a player gameobject to look and move around with.



The parent gameobject, "Player" needs to have the **Character Controller** component, the "**FpsCamera**" & "**FpsMovement**" **scripts** attached to it.
(Capsule collider and Rigidbody are present, so the "**PlayerAvoidance**" on the Flock Manager correctly detects your player gameobject.)

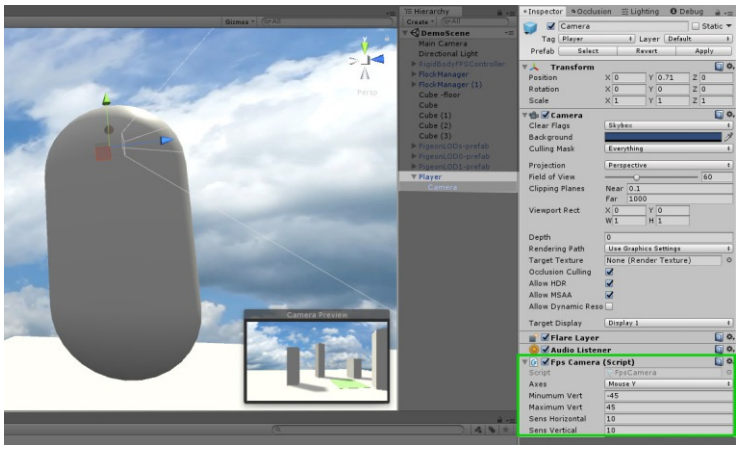
A1. In the "**FpsCamera**" script (on your Player parent object), set the **Axes** to **MouseX**.

A2. The **Minimum Vert** & **Maximum Vert** fields determine the **vertical rotation limits** of your camera.

A3. The **Sens Horizontal** & **Sens Vertical** fields determine the **sensitivity** of your camera's rotation.

B1. In the "**FpsMovement**" script, set the **Speed** parameter to whatever speed you desire your character to move by.

B2. Set the **Gravity** field to a **negative number** (-...) to create the fake gravity, which is "pulling" your character downwards.



The child of the "Player" gameobject, the "Camera" needs to have the **Camera**, **Audio Listener** components and the "**FpsCamera**" **script** attached to it.

In the "**FpsCamera**" script (on your Camera child object), set the **Axes** to **MouseY**.

The fields **Minimum Vert**, **Maximum Vert**, **Sens Horizontal** & **Sens Vertical** should have the same parameters as the ones on the "Player" parent gameobject.