

Hawk Dove Simulation

Due: 19 March

This is a simplification of the Hawk Dove Law Abider problem discussed in class. You will create a Hawk-Dove simulation in Unity 3D following the Hawk-Dove Pseudocode journal article located in Canvas. Recreate their results.

Some setup problems you **may** have:

1. Error code 1617 (Can be ignored):
 - a. In the two files, `Assembly-CSharp.csproj` and `Assembly-CSharp-Editor.csproj`, find the keyword `LangVersion`. Between the tags will be the keyword `"latest"`. Change that to `"Default"`. When you get back to Visual Studio, tell it to reload the project settings.
2. I used TextMeshPro for my UI textfields, buttons and dropdown widgets. It is available in Asset Store for **free**.
3. Use Dynamic Line Chart to track important data. It is **free** in the Asset store.
 - a. See comment by youandi051024 in reviews... recopied here for your help...

1. Import Dynamic Line Chart asset and show some errors
2. Open Package Manager on Unity Editor
3. Show TextMesh Pro and click see all versions button
4. Change TextMesh Pro version 2.x.x to 1.4.1
5. Modify DD_Lines.cs add null checking

@ line 56

```
before: GameObject parent = gameObject.transform.parent.gameObject;
```

```
after: GameObject parent = gameObject.transform.parent ?
        gameObject.transform.parent.gameObject : null;
```

@ line 61

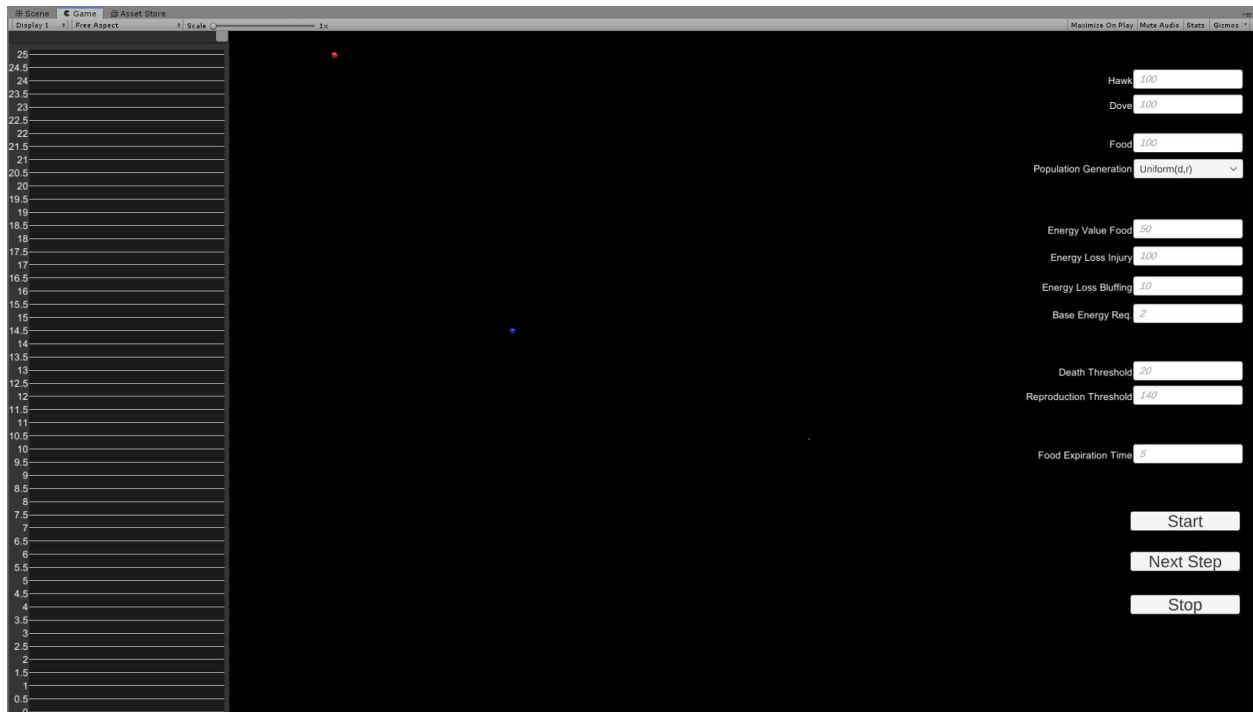
```
before: RectTransform parentrt = parent.GetComponent<RectTransform>();
```

```
after: RectTransform parentrt = parent ? parent.GetComponent<RectTransform>() : null;
```

6. Re-Import Dynamic Line Chart asset exclude DD_Lines.cs
7. finish

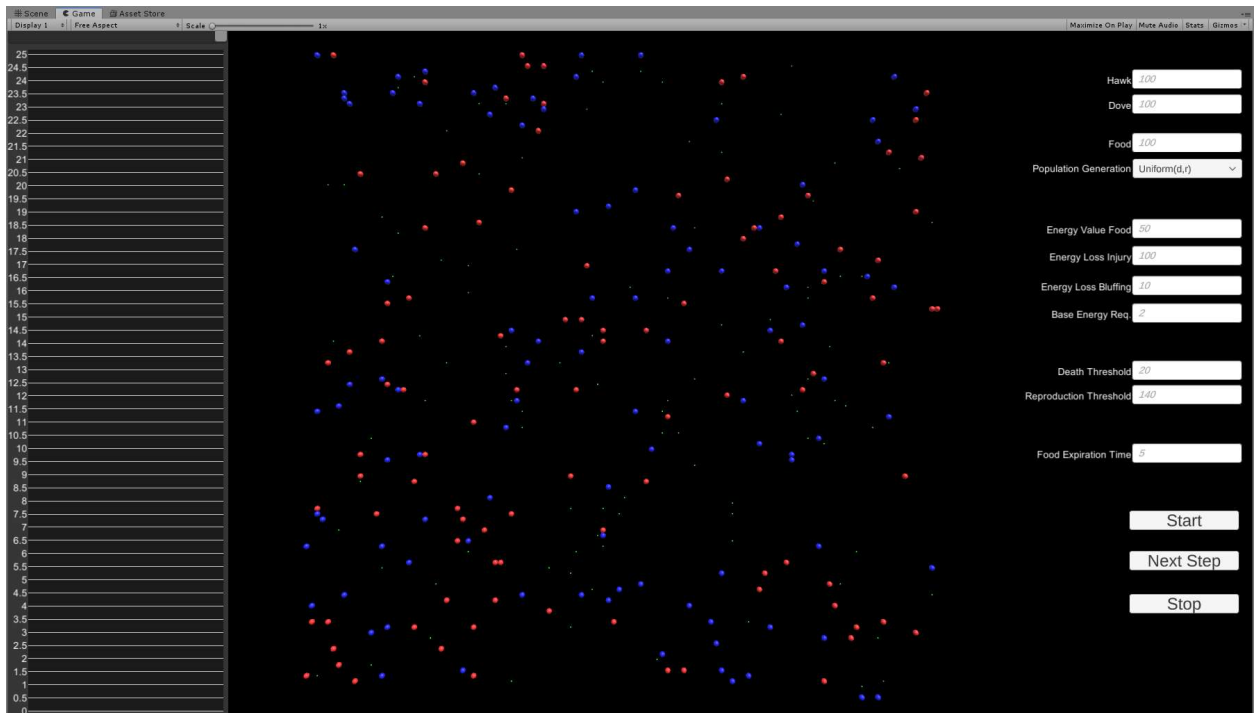
- Size of Panel \rightarrow 320x990. This allows a [0..25] range on y-axis.
- Food/Hawk/Dove must be tracked for 100 generations. At the beginning of each generation, place the data in the chart.

- d. Take values and divide by 10 before displaying on scroll chart. This will handle population growth from [0..250].
 - e. Record data for each run in a file called "runDDMMYYHHMMSS.csv" where DDMMYYHHMMSS stands for Day, Month, Year, Hour, Minute, Second and "csv" means comma separated values. For example, run2402201409.csv would translate to the experiment run at 24 Feb '20 at 4:09 pm.
 - f. This file should be able to be loaded to a player to show the animate the data.
 - g. Since it will be a CSV file, Excel will be able to read it.
4. Create a simulation testbed similar to the following layout. Note the very small Red, Green and Blue balls on the screen.

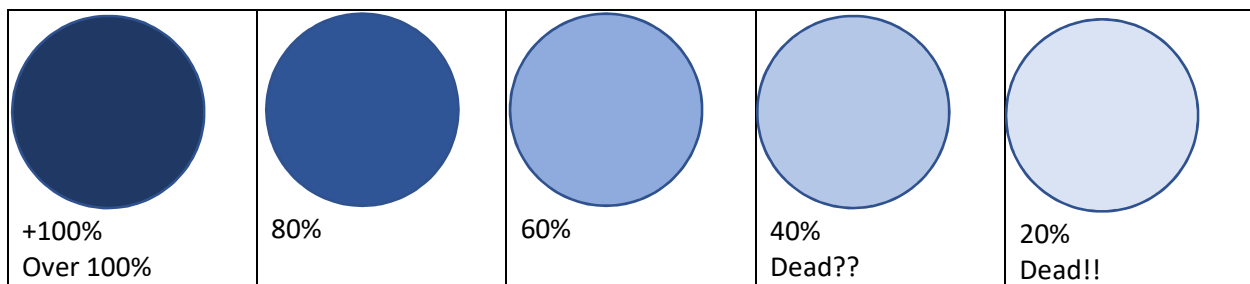


- a. The Red ball represents the Hawk, the Blue ball is the Dove and the Green ball is the Food. They are randomly placed on the screen in a black field of 100x100.
- b. Your UI does not have to look exactly like this. In fact, this UI is just to test the program. Once it is correct, then adding an additional button that runs the entire experiment would be a reasonable addition to the UI.

5. After starting the population, the field will look something like:

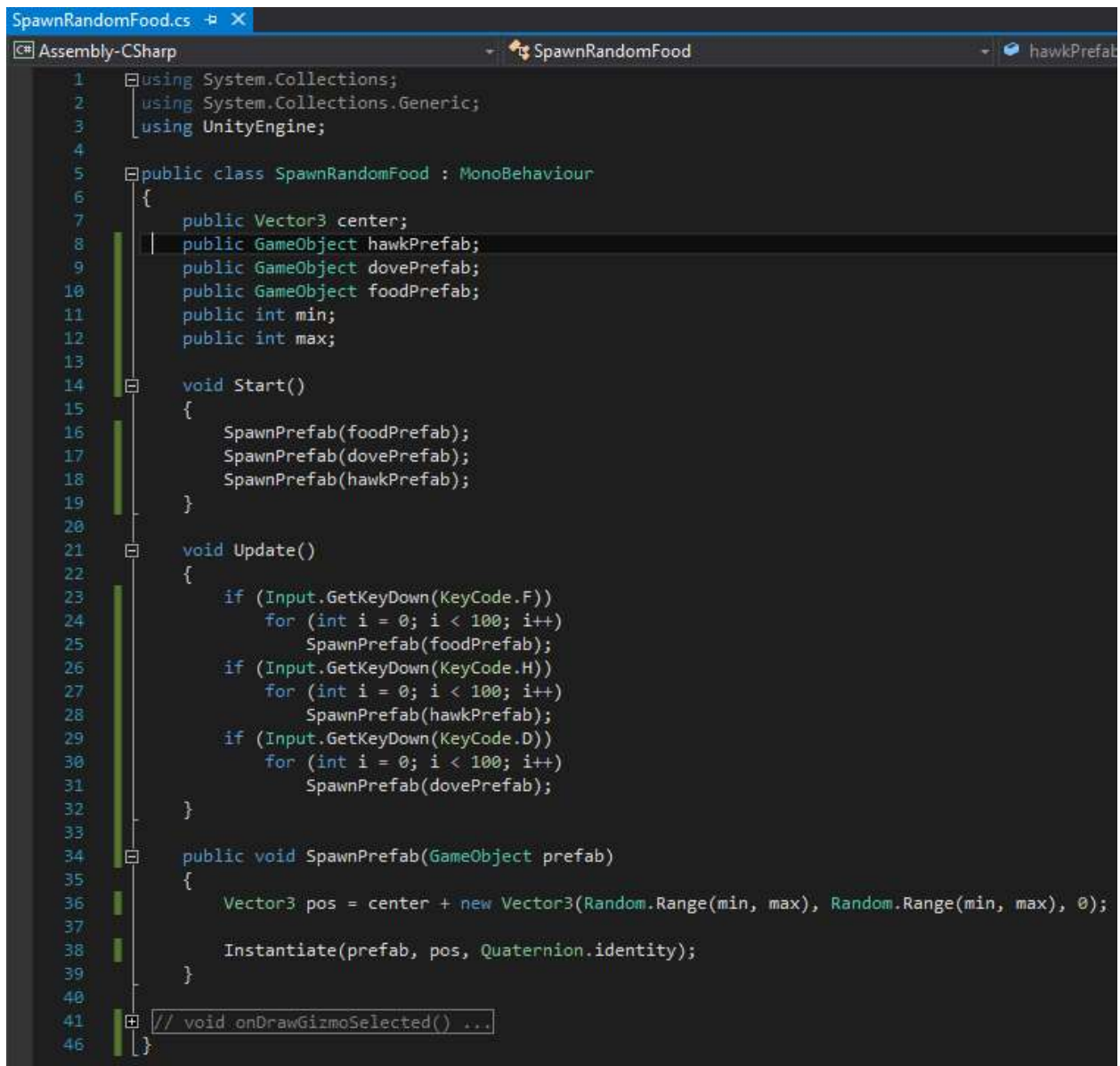


- a. This shows 100 doves, 100 hawks and 100 food units at the start of the experimental run.
 - b. I populated this manually (see following code) using keyboard input. This shows the population at the beginning of the 1st generation. No fights have happened.
6. After each generation, the Doves, Hawks and Food are “aged” according to each object’s rules. As energy is used, the object will fade proportionally. When it gets below the death threshold, it will be discarded. If it gets above the Reproduction Threshold, it creates a child. This child is randomly placed on the field.



- a. For the Dove, the aging looks something like this.
- b. The question to ask, does the 100% map to Reproduction Threshold (Maximum value of a Dove)? Is the Maximum value a cap or can it exceed 140.

7. The code for the randomly spawning of the objects within a specific range is



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class SpawnRandomFood : MonoBehaviour
6 {
7     public Vector3 center;
8     public GameObject hawkPrefab;
9     public GameObject dovePrefab;
10    public GameObject foodPrefab;
11    public int min;
12    public int max;
13
14    void Start()
15    {
16        SpawnPrefab(foodPrefab);
17        SpawnPrefab(dovePrefab);
18        SpawnPrefab(hawkPrefab);
19    }
20
21    void Update()
22    {
23        if (Input.GetKeyDown(KeyCode.F))
24            for (int i = 0; i < 100; i++)
25                SpawnPrefab(foodPrefab);
26        if (Input.GetKeyDown(KeyCode.H))
27            for (int i = 0; i < 100; i++)
28                SpawnPrefab(hawkPrefab);
29        if (Input.GetKeyDown(KeyCode.D))
30            for (int i = 0; i < 100; i++)
31                SpawnPrefab(dovePrefab);
32    }
33
34    public void SpawnPrefab(GameObject prefab)
35    {
36        Vector3 pos = center + new Vector3(Random.Range(min, max), Random.Range(min, max), 0);
37
38        Instantiate(prefab, pos, Quaternion.identity);
39    }
40
41    // void onDrawGizmoSelected() ...
42
43 }
```

- a) It will have to be modified to complete the project. For instance, the color is not modified for color. (Should be added to prefab?) This generates 101 of each object because of Start adding one each. (Start should automatically generate 100 of each object, while Update should only generate 100 food.)

8. Use the pseudocode algorithm in the paper to create a simulation. Each Run of the experiment will have 100 iterations.

```

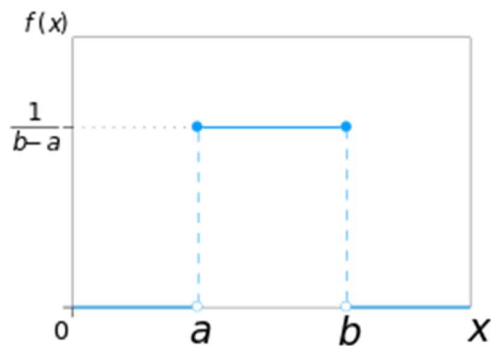
Remove expired food
Add new food
For each agent
    If the agent has not yet competed in this time period
        If food is available
            If another agent wants the food
                Compete for the food and update Energy
            Else
                Take the food and update Energy
        Subtract base from Energy
    If Energy < d
        Remove the agent from the population
    Else if Energy > r
        Create an offspring with Energy/2 initial energy
        and reduce this agent's Energy to Energy/2

```

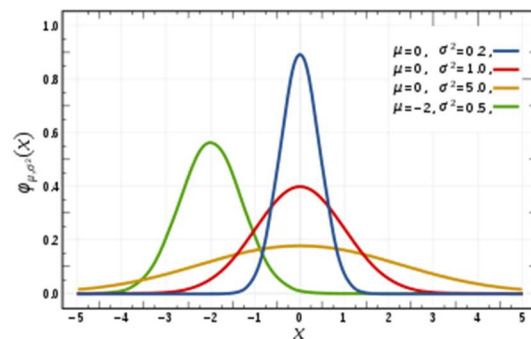
9. Note that the default values in the menu are from the paper.

- Initial population: 100 hawks and 100 doves
- Energy value of food items: $v = 50$ units
- Energy loss for injury: $i = 100$ units
- Energy loss for bluffing: $b = 10$ units
- Death threshold: $d = 20$ units
- Reproduction threshold: $r = 140$ units
- Initial energy level for each hawk or dove: sampled uniformly from (d, r)
- Base energy requirement per time period: $base = 2$ units
- Time periods before a food item expires: 5
- Food supply: 100 new food items per time period

- a) The “Initial energy requirement per time period” has the “sampled uniformly from (d, r) ” option chosen. This should also have other options, such as “sampled normally from (d, r) ” and any others mentioned in paper.



a. Uniform distribution



b. Normal distribution

b) Note that the Normal distribution may need more input than just (d, r)