

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
ĐẠI HỌC QUỐC GIA HÀ NỘI



MÔ PHÒNG HIỆN TƯỢNG BẦY ĐÀN

Giảng viên: Nguyễn Hồng Thịnh

Nhóm thực hiện:

Vũ Trung Đức - 21021577

Nguyễn Văn Hiệp - 21021582

Bùi Duy Hoàng Anh - 21021552

Vũ Phương Nhi - 21021622

Mục lục

1	Tổng quan	2
2	Giới thiệu	3
3	Các tác động	4
3.1	Wander - sự ngẫu nhiên	4
3.2	Cohesion - sự gắn kết	4
3.3	Alignment - cân chỉnh	5
3.4	Separation - tách biệt	6
3.5	Avoidance - tránh né	8
3.6	Combine - tổng hợp tác động	8
4	Xử lý UI	9
5	Một số hình ảnh ứng dụng	10
6	Kết Luận	12

1 Tổng quan

Một ứng dụng Unity2D mô phỏng hiện tượng bầy đàn đã được phát triển. Ứng dụng này cho phép người dùng khám phá và tìm hiểu về các tác động quan trọng trong hiện tượng bầy đàn, bao gồm wander (lang thang), cohesion (gắn kết), alignment (cân chỉnh), separation (tách biệt) và avoidance (tránh xa).

Trong chương trình, người dùng sẽ được chứng kiến sự di chuyển của một đám đàn bầy gồm nhiều cá thể (hình tròn) trên màn hình. Mỗi cá thể được điều khiển bởi các thuật toán mô phỏng các tác động bầy đàn.

Tác động wander cho phép các cá thể di chuyển ngẫu nhiên trong không gian, tạo ra sự linh hoạt và sự ngẫu hứng trong cách chúng di chuyển.

Tác động cohesion tạo ra sự gắn kết giữa các cá thể trong bầy. Điều này khiến chúng tụ lại gần nhau và hình thành một cấu trúc đồng nhất.

Tác động alignment đảm bảo rằng các cá thể trong bầy di chuyển theo cùng một hướng hoặc hướng tương tự nhau. Điều này tạo ra sự tổ chức và đồng bộ trong cách chúng di chuyển.

Tác động separation đảm bảo rằng các cá thể trong bầy không giao nhau hoặc không tiếp xúc quá gần. Điều này tránh va chạm và xung đột không mong muốn giữa các cá thể.

Tác động avoidance đảm bảo rằng các cá thể trong bầy có khả năng tránh né hoặc không tiếp xúc quá gần kẻ thù.

Người dùng có thể tương tác với chương trình bằng cách bật tắt chức năng và quan sát sự thay đổi trong hành vi của bầy đàn. Điều này giúp họ hiểu rõ hơn về cách các tác động này ảnh hưởng đến sự tổ chức và di chuyển của bầy đàn.

Chương trình mô phỏng hiện tượng bầy đàn của tôi không chỉ mang tính giáo dục mà còn mang tính giải trí. Nó cung cấp cho người dùng một cách thú vị để khám phá và tìm hiểu hiện tượng phức tạp của sự tổ chức và tương tác trong một bầy đàn.

Tôi hy vọng rằng chương trình này sẽ giúp người dùng hiểu rõ hơn về hiện tượng bầy đàn và tạo ra sự quan tâm và tò mò trong lĩnh vực này.

2 Giới thiệu

Flocking là một hành vi mà một tập hợp các thực thể tự thúc đẩy độc lập di chuyển cùng nhau một cách tập thể [1]. Để tham gia vào một đàn, mỗi thực thể phải có khả năng điều phối chuyển động của riêng mình với chuyển động của các thực thể hàng xóm. Một thực thể không thể di chuyển quá nhanh hoặc quá chậm so với đàn; nó cũng không thể quá gần hoặc quá xa so với bất kỳ thực thể hàng xóm gần nhất nào. Mỗi thực thể phải liên tục điều chỉnh chuyển động của mình để không làm gián đoạn bất kỳ thực thể hàng xóm nào khác. Hành vi đàn có thể được tìm thấy cả trong môi trường tự nhiên và nhân tạo. Các đàn tự nhiên phổ biến bao gồm các đàn chim hoặc đàn cá. Hành vi đàn nhân tạo thường được tìm thấy trong robot và hàng không vũ trụ.

3 Các tác động

3.1 Wander - sự ngẫu nhiên

Các bước tiến hành:

- Tính toán một giá trị "jitter - độc lập" dựa trên một tham số "wander jitter" và thời gian giữa các khung hình.
- Cập nhật mục tiêu di chuyển ngẫu nhiên bằng cách thêm một vector ngẫu nhiên với biên độ bị ảnh hưởng bởi "jitter".
- Chuẩn hóa mục tiêu di chuyển để chỉ định hướng di chuyển mà không quan tâm đến khoảng cách.
- Điều chỉnh độ dài của mục tiêu di chuyển để nằm trong một phạm vi bán kính xác định.
- Chuyển đổi mục tiêu từ không gian cục bộ sang không gian toàn cầu.
- Trả về mục tiêu di chuyển chuẩn hóa.

```
38 protected Vector3 Wander()  
39 {  
40     float jitter = config.wander_jitter * Time.deltaTime;  
41     wander_target += new Vector3(RandomBinomial() * jitter, RandomBinomial() * jitter, 0);  
42     wander_target = wander_target.normalized;  
43     wander_target *= config.wander_radius;  
44  
45     Vector3 target_in_local_space = wander_target + new Vector3(0, config.wander_distance, 0);  
46     Vector3 target_in_world_space = transform.TransformPoint(target_in_local_space);  
47     return target_in_world_space.normalized;  
48 }
```

Hình 1: Thực hiện code C#

3.2 Cohesion - sự gắn kết

Cơ sở lý thuyết:

$$V_i(k+1) = \frac{\sum_{n \in N} P_n(k)}{N} - P_i(k)$$

Trong đó:

- $V_i(k+1)$ là vector vận tốc.

- N là số lượng thành viên trong tầm nhìn.
- $P_n(k)$ là giá trị vị trí của thành viên.

Các bước tiến hành:

- Tìm kiếm những đối tượng nằm trong phạm vi.
- Nếu không có đối tượng nào trả về vector 0, ngược lại tiến hành tiếp bước 3
- Duyệt danh sách các đối tượng trong phạm vi và tổng hợp những đối tượng nằm trong vùng nhìn.
- Nếu số lượng trong phạm vi khác 0 giảm độ lớn của vector tổng hợp sau đó tính toán vector chênh lệch với vị trí hiện tại.
- Trả về vector di chuyển chuẩn hóa.

```

49 Vector3 Cohesion()
50 {
51     Vector3 cohesion_vector = new Vector3();
52     int count_members = 0;
53     var neighbors = level.GetNeighbors(this, config.conhesion_radius);
54     if(neighbors.Count == 0)
55     {
56         return cohesion_vector;
57     }
58     foreach(var neighbor in neighbors)
59     {
60         if (IsInFOV(neighbor.positon))
61         {
62             cohesion_vector += neighbor.positon;
63             count_members++;
64         }
65     }
66     if(count_members == 0)
67         return cohesion_vector;
68     cohesion_vector /= count_members;
69     cohesion_vector = cohesion_vector - this.positon;
70     cohesion_vector = Vector3.Normalize(cohesion_vector);
71     return cohesion_vector;
72 }

```

Hình 2: Thực hiện code C#

3.3 Alignment - cân chỉnh

Cơ sở lý thuyết:

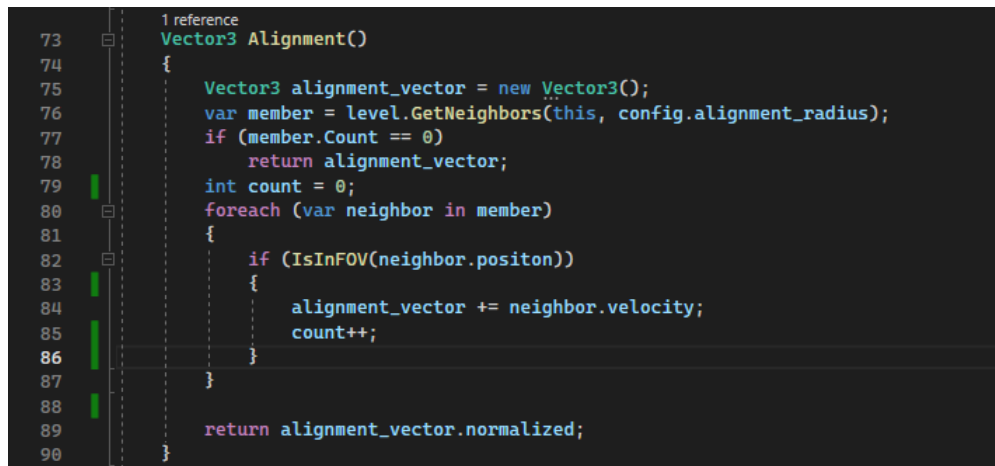
$$V_i(k+1) = \frac{\sum_{n \in N} V_n(k)}{N}$$

Trong đó:

- $V_i(k + 1)$ là vector vận tốc.
- N là số lượng cá thể trong vùng xét.
- $V_n(k)$ là giá trị vận tốc của thành viên lân cận.

Các bước tiến hành:

- Tìm kiếm những đối tượng nằm trong phạm vi.
- Nếu không có đối tượng nào trả về vector 0, ngược lại tiến hành tiếp bước 3
- Duyệt danh sách các đối tượng trong phạm vi và tổng hợp những đối tượng nằm trong vùng nhìn. Nếu đối tượng trong vùng nhìn tiến hành cộng vector tốc độ (neighbor.velocity) của thành viên lân cận vào alignmentVector.
- Trả về vector hướng di chuyển đã chuẩn hóa.



```

73 1 reference
74  Vector3 Alignment()
75  {
76      Vector3 alignment_vector = new Vector3();
77      var member = level.GetNeighbors(this, config.alignment_radius);
78      if (member.Count == 0)
79          return alignment_vector;
80      int count = 0;
81      foreach (var neighbor in member)
82      {
83          if (IsInFOV(neighbor.position))
84          {
85              alignment_vector += neighbor.velocity;
86              count++;
87          }
88      }
89      return alignment_vector.normalized;
90  }

```

Hình 3: Thực hiện code C#

3.4 Separation - tách biệt

Cơ sở lý thuyết:

$$V_i(k + 1) = \frac{\sum_{n \in N} \left(\frac{P_i(k) - P_n}{|P_i(k) - P_n|} \right)}{N}$$

- $V_i(k + 1)$ là separateVector (vector tách biệt) của đối tượng thứ i tại bước thời gian $k + 1$.

- $P_i(k)$ là vị trí của đối tượng thứ i tại thời điểm k .
- P_n là vị trí của đối tượng hàng xóm thứ n .
- $|P_i - P_n|$ là khoảng cách giữa vị trí của đối tượng thứ i và đối tượng hàng xóm thứ n .
- N là số lượng đối tượng hàng xóm trong tập N .

Các bước tiến hành:

- Tìm kiếm những đối tượng nằm trong phạm vi.
- Nếu không có đối tượng nào trả về vector 0, ngược lại tiến hành tiếp bước 3
- Duyệt danh sách các đối tượng trong phạm vi và tổng hợp những đối tượng nằm trong vùng nhìn. Nếu đối tượng trong vùng nhìn tính toán vector movingTowards bằng hiệu của vị trí hiện tại (this.positon) và vị trí của thành viên lân cận (neighbor.positon), biểu thị hướng di chuyển từ đối tượng hiện tại đến thành viên lân cận.
- Kiểm tra nếu độ lớn của vector movingTowards lớn hơn 0 (movingTowards.magnitude > 0), tức là đối tượng hiện tại và thành viên lân cận không trùng vị trí, thì thực hiện phân tách bằng cách cộng vector chuẩn hóa (movingTowards.normalized) chia cho độ lớn của movingTowards (movingTowards.magnitude) vào separateVector.
- Trả về vector hướng di chuyển đảm bảo phân cách giữa các đối tượng trong vùng nhìn đã chuẩn hóa.

```

86  Vector3 Separation()
87  {
88      Vector3 separate_vector = new Vector3();
89      var member = level.GetNeighbors(this, config.alignment_radius);
90      if (member.Count == 0)
91          return separate_vector;
92      foreach (var neighbor in member)
93      {
94          if (IsInFOV(neighbor.positon))
95          {
96              Vector3 moving_towards = this.positon - neighbor.positon;
97              if (moving_towards.magnitude > 0)
98              {
99                  separate_vector += moving_towards.normalized / moving_towards.magnitude;
100              }
101          }
102      }
103      return separate_vector.normalized;
104  }

```

Hình 4: Thực hiện code C#

3.5 Avoidance - tránh né

Các bước tiến hành:

- Tìm kiếm những đối tượng kẻ thù nằm trong phạm vi.
- Nếu không có đối tượng nào trả về vector 0, ngược lại tiến hành tiếp bước 3
- Duyệt danh sách các đối tượng kẻ thù trong phạm vi và vector hướng cần thiết để chạy tránh kẻ thù. Trong hàm RunAway thực hiện tính toán độ lớn hướng cần di chuyển với tốc độ tối đa. Rồi trả về độ chênh lệch vận tốc cần chạy và vận tốc hiện tại.
- Trả về vector hướng di chuyển tổng hợp để tránh các đối tượng thù địch đã được chuẩn hóa.

```
104 }
105 Vector3 Avoidance()
106 {
107     Vector3 avoidance = new Vector3();
108     var enemyList = level.GetEnemies(this, config.avoidane_radius);
109     if (enemyList.Count == 0)
110         return avoidance;
111     foreach (var enemy in enemyList)
112     {
113         avoidance += RunAway(enemy.positon);
114     }
115     return avoidance.normalized;
116 }
117 Vector3 RunAway(Vector3 target)
118 {
119     Vector3 needed_velocity = (positon - target).normalized * config.max_velocity;
120     return needed_velocity - velocity;
121 }
```

Hình 5: Thực hiện code C#

3.6 Combine - tổng hợp tác động

Trả về vector tổng hợp hướng từ các tác động gây ra, với regime là một mảng giá trị 0/1 (tắt/bật) tác động.

```

122     virtual protected Vector3 Combine()
123     {
124         Vector3 final_vec = new Vector3();
125         final_vec += config.conhesion_priority * Cohesion() * level.regime[0]
126                   + config.wander_priority * Wander() * level.regime[1]
127                   + config.alignment_priority*Alignment() * level.regime[2]
128                   + config.separation_priority*Separation() * level.regime[3]
129                   + config.avoidane_priority*Avoidance() * level.regime[4];
130         return final_vec;
131     }

```

Hình 6: Thực hiện code C#

4 Xử lý UI

Sử dụng tính năng UI của Unity như slider (thanh trượt), button (nút bấm), check box (lựa chọn), Text (chuỗi hiển thị), ... kết hợp với logic ứng dụng ta có thể điều chỉnh được tốc độ mô phỏng, số lượng cá thể mô phỏng, các tác nhân ảnh hưởng.

5 Một số hình ảnh ứng dụng



Hình 7: Màn hình bắt đầu



Hình 8: Màn hình điều hướng



Hình 9: Màn hình mô phỏng



Hình 10: Màn hình mô phỏng

6 Kết Luận

Trên cơ sở xây dựng ứng dụng mô phỏng hiện tượng bầy đàn trong môi trường Unity2D, chúng tôi đã đạt được những kết quả đáng chú ý. Dưới đây là một số điểm quan trọng và kết luận từ quá trình phát triển và thử nghiệm ứng dụng:

1. Tính Tương Tác và Hiện Thực:

- Ứng dụng đã thành công trong việc mô phỏng các tác động quan trọng trong hiện tượng bầy đàn như wander, cohesion, alignment, separation và avoidance.
- Người dùng có khả năng quan sát và tương tác với bầy đàn, bật/tắt từng tác động để hiểu rõ hơn về ảnh hưởng của chúng đối với tổ chức và di chuyển của bầy đàn.

2. Tính Giáo Dục và Giải Trí:

- Ứng dụng không chỉ mang lại trải nghiệm giáo dục về hiện tượng bầy đàn mà còn mang lại sự giải trí cho người dùng.
- Việc kết hợp giáo dục và giải trí giúp tạo ra một cách tiếp cận thú vị và hấp dẫn, khuyến khích sự tò mò và quan tâm trong lĩnh vực này.

3. Mở Rộng và Phát Triển Tiềm Năng:

- Ứng dụng cung cấp cơ hội cho sự mở rộng và phát triển trong tương lai. Có thể thêm vào các tính năng mới, tăng cường đồ họa, hoặc thậm chí tích hợp với các nền tảng khác để mở rộng ảnh hưởng và sử dụng của ứng dụng.

4. Khám Phá Sâu Hơn:

- Chúng tôi hy vọng rằng ứng dụng này sẽ khuyến khích người dùng tiếp tục khám phá sâu hơn về hiện tượng bầy đàn và tạo ra sự quan tâm trong lĩnh vực nghiên cứu và ứng dụng.

Tổng cộng, dự án mô phỏng hiện tượng bầy đàn của chúng tôi không chỉ là một công cụ giáo dục hữu ích mà còn là một trải nghiệm giải trí độc đáo. Chúng tôi kỳ vọng rằng ứng dụng này sẽ đóng góp vào việc tăng cường hiểu biết và sự quan tâm đối với lĩnh vực đa dạng và phức tạp của hành vi bầy đàn.