SIRSモデル 加筆部分まとめ

Boidシミュレーションへの SIRS（感染モデル）拡張 ― 3ファイルの加筆箇所（#SIRS）を抽出・整理

1. SingleBoid.cs

各ボイド個体に「感染状態」を持たせるための変数とメソッド。クラス末尾（接触距離の設定メソッドの後）に追記。

▶ 加筆位置：クラス末尾に追加

```
/* SIRS関係の変数 （6月23日）*/
public bool infection = false;    //感染の有無
public bool susceptible = true;   //感染能力の有無
public float timeI = 0f;          //感染後の経過時間（秒）
public float timeR = 0f;          //回復後の経過時間（秒）

// 感染状態をリセットする
public void ResetInfection(){
    infection = false;     //感染を無くし、
    susceptible = true;    //免疫も無くす
    timeI = 0f;
    timeR = 0f;
}

// 現在の感染状態を返す関数
public string SIRS(){

    /* （免疫を失い）感染能力の有る状態（Susceptible） */
    if(!infection && susceptible){
        return "S";
    }
    /* 感染している状態（Infection） */
    if(infection){
        return "I";
    }
    /* 感染から回復し免疫のある状態（Recovery） */
    if(!infection && !susceptible){
        return "R";
    }

    return "";
}
```

2. BoidRuleManager.cs

SIRS状態遷移（S→I→R→S）のルール本体。フィールド宣言部と、ルール適用メソッド群の最後に追記。

▶ 加筆位置：フィールド宣言部 ＋ クラス内のメソッドとして追加

```
/* SIRSモード */
public float time_ItoR = 10f; //最大感染持続時間 [秒]
public float time_RtoS = 40f; //最大免疫持続時間 [秒]

// SIRSモデルを適用する.
public void ApplySIRS(){

    for(int i=0; i<pop; i++){

        //ボイド[i]の位置
        Vector3 ipos = boid[i].pos;
        //ボイド[i]の近接距離
        float ineighbor_space = boid[i].neighbor_space;

        /*********************************
          （1）ボイド[i]がS状態のとき
        *********************************/

        //近接位置にいるボイド[j]がI状態の時、
        //確率1/20でボイド[i]もI状態へ

        if(boid[i].SIRS().Contains("S")){

            for(int j=0; j<pop; j++){

                Vector3 jpos = boid[j].pos;
                float dis_ij = Vector3.Distance(ipos, jpos);

                if( i!=j && dis_ij<ineighbor_space && boid[j].infection){
                    if(Random.value < 0.05f){
```

```
                        boid[i].infection = true;
                        boid[i].timeI = 0f;
                        break;
                    }
                }
            }
        }

        /*********************************
          （2）ボイド[i]がI状態のとき
        *********************************/

        // I状態になってからの経過時間が規定時間（time_ItoR）を超えたら、
        // 確率1/20でR状態へ

        if(boid[i].SIRS().Contains("I")){

            boid[i].timeI += Time.deltaTime;

            if(boid[i].timeI > time_ItoR){
                if(Random.value < 0.05f){
                    boid[i].infection = false;
                    boid[i].susceptible = false;
                    boid[i].timeR = 0f;
                }
            }
        }

        /*********************************
          （3）ボイド[i]がR状態のとき
        *********************************/

        // I状態になってからの経過時間が規定時間（time_RtoS）を超えたら、
        // 確率1/20でS状態へ

        if(boid[i].SIRS().Contains("R")){

            boid[i].timeR += Time.deltaTime;

            if(boid[i].timeR > time_RtoS){
                if(Random.value < 0.05f){
                    boid[i].susceptible = true;
                    boid[i].infection = false;
                }
            }
        }
    }
}
```

3. BoidManager.cs

既存コードの各所に散在する加筆。挿入位置ごとに分けて示す（各行の「#SIRS / 追記」コメントが加筆目印）。

3-1. フィールド宣言部

▶ ボイド配列 boid[] の宣言の直後

```
private GameObject[] boidobj; //追記 #SIRS
```

3-2. Start() 内（ボイド生成ループ）

▶ boidobj 配列を確保し、生成した GameObject を保持

```
GameObject bpar = GameObject.Find("ParentBoid");
boidobj = new GameObject[pop];        //追記 #SIRS
for (int i = 0; i < pop; i++)
{
    GameObject bobj = Instantiate((GameObject)Resources.Load("Boid"), bpar.transform);
    boid[i] = bobj.GetComponent<SingleBoid>();
    boidobj[i] = bobj;                 //追記 #SIRS
}
```

3-3. Update() 内

▶ ルール適用（ApplyRules）の後に SIRS の適用と可視化を追加

```
//SIRSルールの適用 #SIRS
rule.ApplySIRS();
ShowInfection();
```

3-4. Iキー処理内（位置・速度の初期化）

▶ 既存の初期化処理に続けて感染状態のリセットを追加

```
if (Input.GetKeyDown(KeyCode.I))
{
    // …既存の初期化処理…

    /* ボイドの感染状態をリセットする #SIRS */
    ResetBoidInfection();
}
```

3-5. Sキー処理（新規）

▶ キーイベント部に新しい分岐を追加

```
/* Sボタンでランダムに感染を発生させる #SIRS */
if (Input.GetKeyDown(KeyCode.S))
{
    this.setRandomInfection();
}
```

3-6. 新規メソッド：ResetBoidInfection()

▶ 全ボイドの感染状態をリセット

```
/* 全てのボイドの感染状態をリセットする #SIRS */
private void ResetBoidInfection()
{
    for (int i = 0; i < pop; i++)
    {
        boid[i].ResetInfection();
    }
}
```

3-7. 新規メソッド：ShowInfection()

▶ 状態に応じて色を変えて可視化（S=赤紫 / I=黄 / R=白）。大きさは全状態 1倍

```
/* 感染状態の可視化 #SIRS */
private void ShowInfection()
{
    for (int i = 0; i < pop; i++)
    {
        Renderer r = boidobj[i].GetComponent<Renderer>();

        if (boid[i].SIRS().Contains("S"))
        {
            r.material.color = new Color(1f, 0f, 0.31f);
            boidobj[i].transform.localScale = new Vector3(1f, 1f, 1f);
        }
        if (boid[i].SIRS().Contains("I"))
        {
            r.material.color = Color.yellow;
            boidobj[i].transform.localScale = new Vector3(1f, 1f, 1f);
        }
        if (boid[i].SIRS().Contains("R"))
        {
            r.material.color = new Color(1f, 1f, 1f);
            boidobj[i].transform.localScale = new Vector3(1f, 1f, 1f);
        }
    }
}
```

3-8. 新規メソッド：setRandomInfection()

▶ 各ボイドを 1% の確率で感染させる

```
/* 感染をランダムに発生させる（1%の確率） #SIRS */
private void setRandomInfection()
{
    for (int i = 0; i < pop; i++)
    {
        if (Random.value < 0.01)
        {
            boid[i].infection = true;
        }
    }
}
```