演習２：集合の知性を設計する

(05)　05/15

Ａ｜Unity環境の整備・簡単なルールの設計

(06) 05/22　(07)　05/29

Ｂ｜ボイドルール１・２・３の実装

(08)　06/05　(09)　06/12

Ｃ｜課題１：集合知の解析

(10)　06/19

Ｄ１｜ＳＩＲ（感染モデル）

(11)　06/26　(12)　07/03　(13)　07/10

Ｄ２｜課題２：マイルール・感染ルール・視点操作

(14-15)　07/17

Ｄ３｜発表（One-Minute Movie）

3つの修正

(1) ルール３（整列）において、ルールの適用前後でボイドの速度を不変とします。

(2) ボイド同士が透過するようにします。

(3) 壁への衝突直後の一定時間にルールの適用をキャンセルします。

3つの修正

(1) ルール3（整列）において、ルールの適用前後でボイドの速度を不変とします。

(2) ボイド同士が透過するようにします。

(3) 壁への衝突直後の一定時間にルールの適用をキャンセルします。

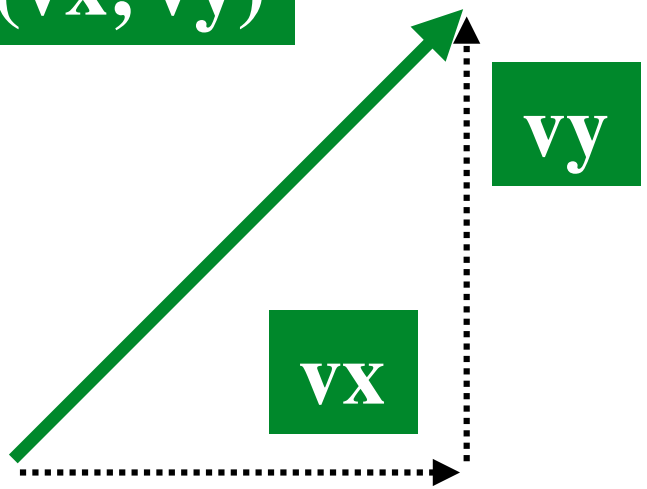
ルール3の修正！

速度を, 別の速度（マスターベクトル）に徐々に向けるためのベクトル計算

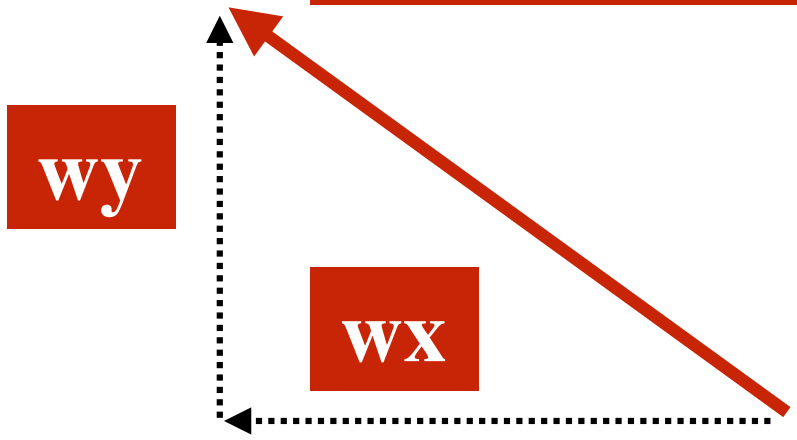
速度による引き込み
（内分ベクトルによる方法）

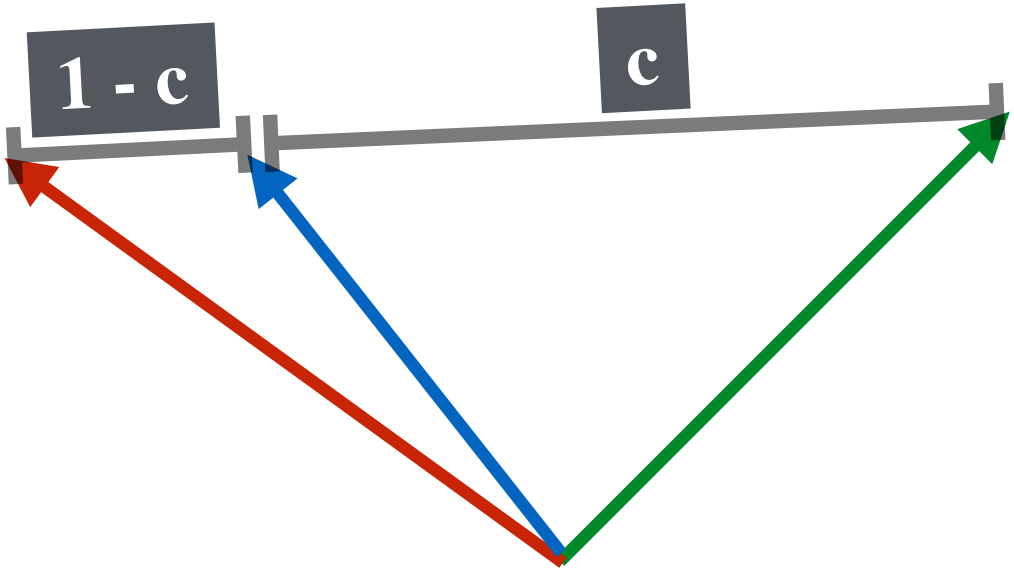
内分ベクトルの計算

v’ = (vx, vy)



w’ = (wx, wy)





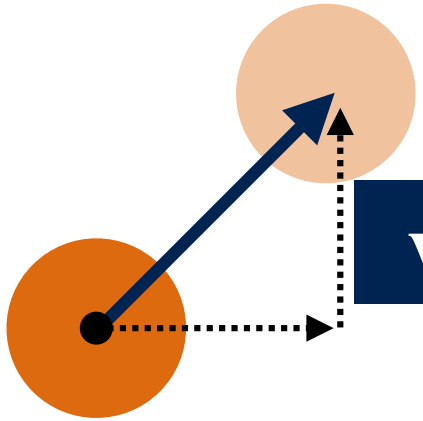
v’とw’を c:1-c に内分するベクトル

(c*wx + (1 - c)*vx, c*wy + (1 - c)*vy)

<速度 v'> を<速度w'> の方向に修正する（内分ベクトルによる方法）

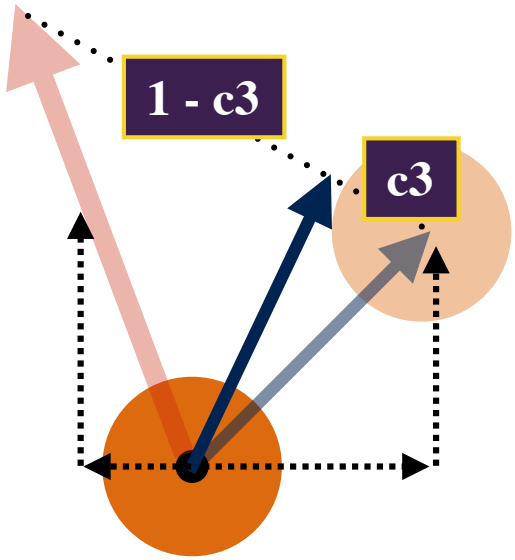
マスターベクトル

w' = (wx, wy)



もともとの
速度ベクトル（v0）

v0' = (vx, vy)

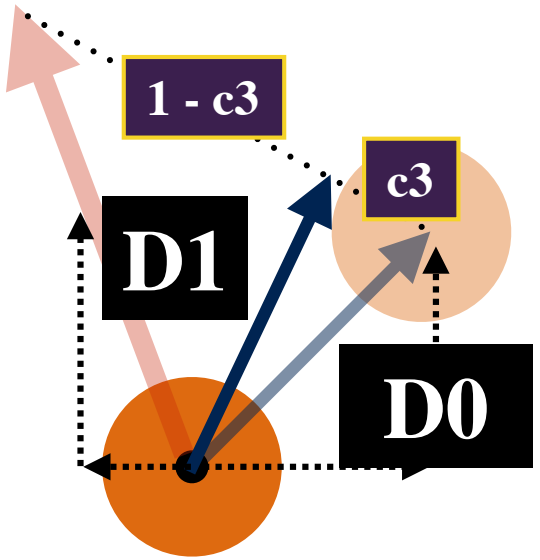


更新された速度ベクトル（v1）

v1' = (c3*wx+(1-c3)*vx,
c3*wy+(1-c3)*vy)

c3 が **1** に近い程, すぐマスターベクトルに引きこまれる.

c3 が **1** に近い程, すぐマスターベクトルに引きこまれる.



更新された速度ベクトル（v1）

$$v1' = (c3*wx+(1-c3)*vx,\ c3*wy+(1-c3)*vy)$$

速さを調整した速度ベクトル（v2）

$$v2' = (D0 / D1) * v1'$$

D0 ：v0’の距離

D1 ：v1’の距離

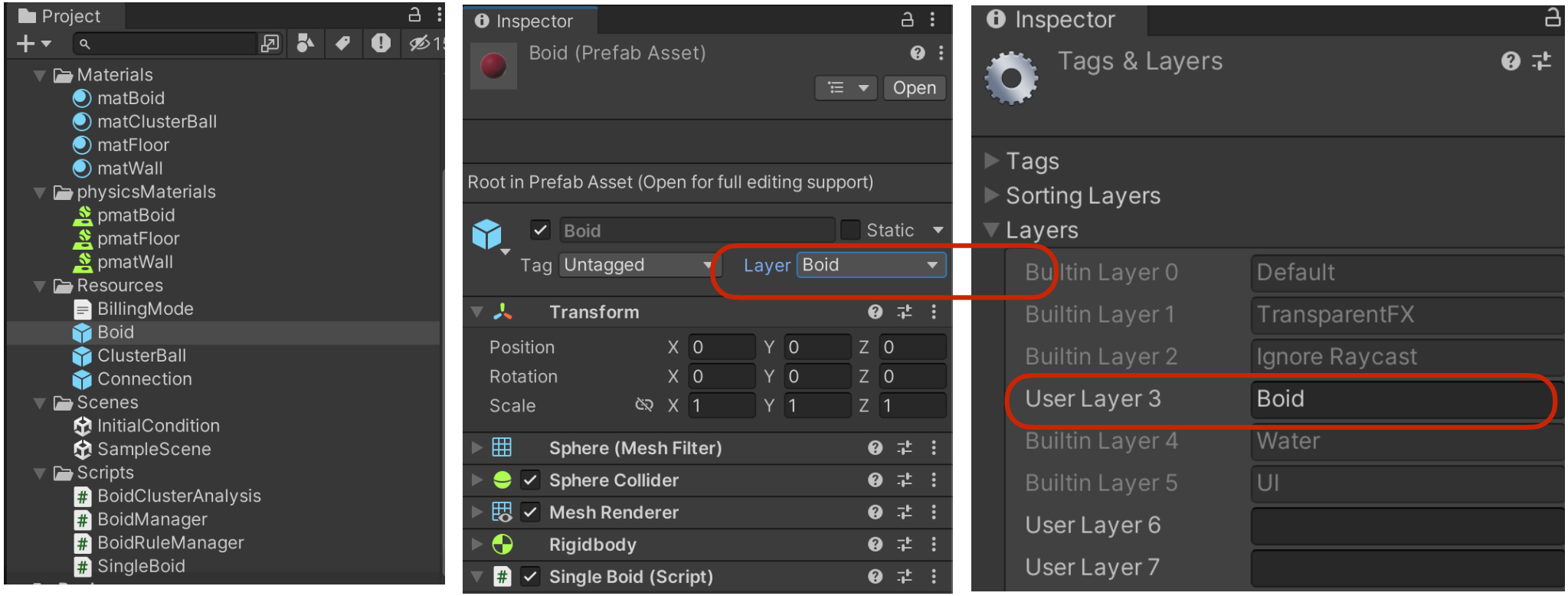
速度ベクトルの距離（速さ）を、D0（もともとの速さ）に調整する。

3つの修正

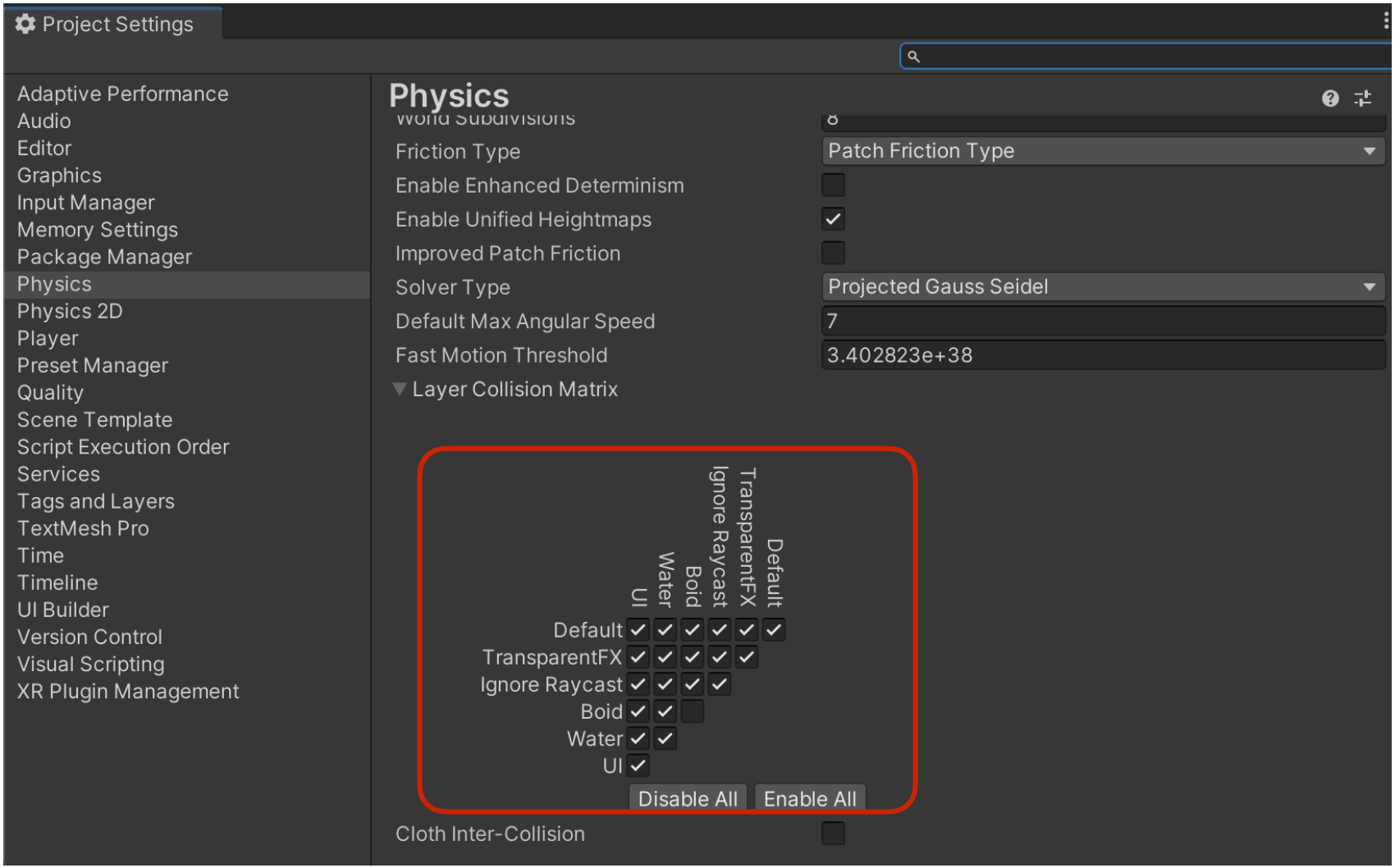
(1) ルール３（整列）において、ルールの適用前後でボイドの速度を不変とします。

(2) ボイド同士が透過するようにします。

(3) 壁への衝突直後の一定時間にルールの適用をキャンセルします。



- 新たに「Boid」という名前のユーザ定義のレイヤを登録し、Builtin Layerの2と4の間に差し込みます。
- プレハブのボイドのレイヤを「Boid」に設定します。



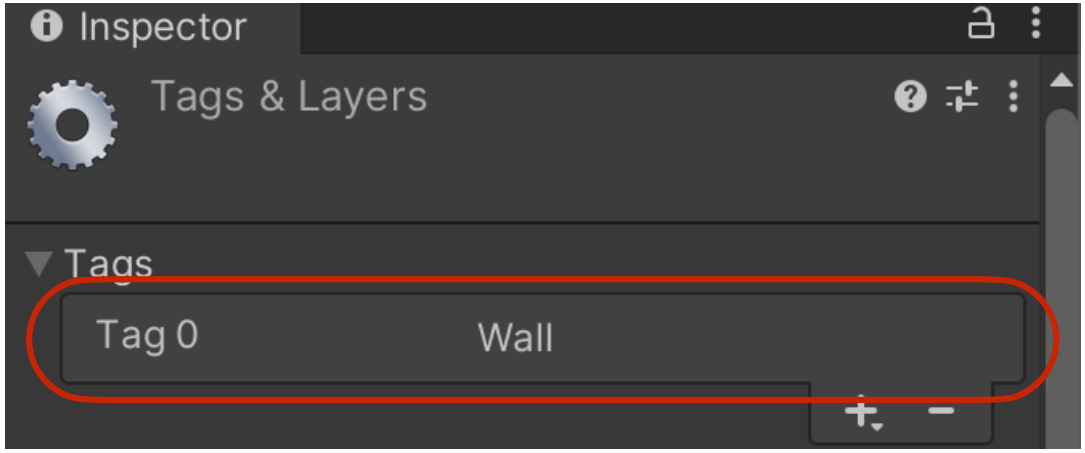
- メニューから「Edit→Project Settings→Physics」で一番下にスクロールすると、レイヤ間のコリジョンの発生の有無を指定できます。ここでは、ボイド同士のコリジョンをオフにします。

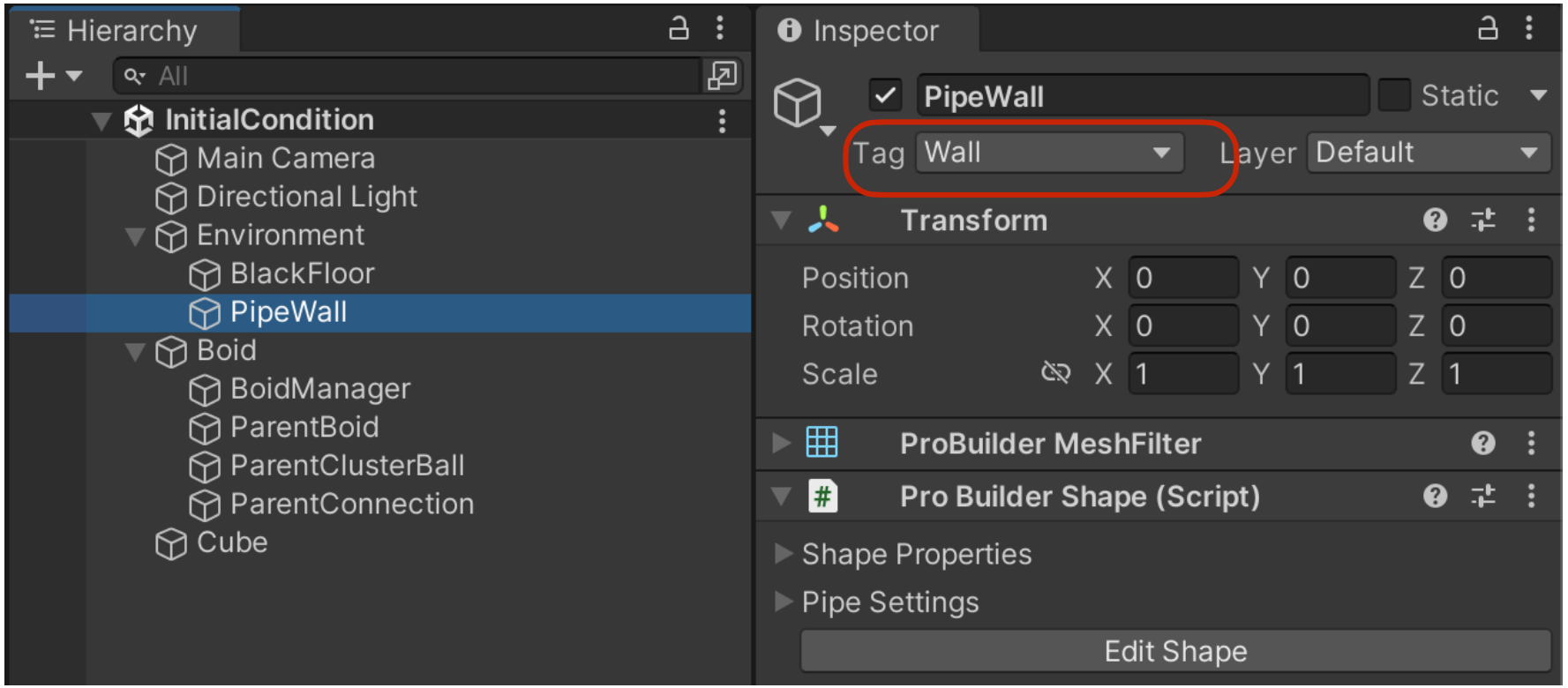
3つの修正

(1) ルール３（整列）において、ルールの適用前後でボイドの速度を不変とします。

(2) ボイド同士が透過するようにします。

(3) 壁への衝突直後の一定時間にルールの適用をキャンセルします。





- 新たに「Wall」という名前のタグを登録しPipeWallのタグとして割り当てます。

SingleBoid.cs

```
12      /* 壁への衝突 x 相互作用の無効 （2024.6.4）*/
13
14      //相互作用の無効（trueのときsetVelocityを機能させない）
        3 個の参照
15      public bool blindness = false;
16
17      //壁に衝突してからの時間（sec）
        3 個の参照
18      private float timeW = 0f;
19
20      //衝突後ボイドルールを無効にする時間（sec）
        1 個の参照
21      private float blindtime_after_wallhit = 0.8f;
22
```

宣言部

Update()

```
        0 個の参照
43      void Update ()
44      {
45          //位置と速度の更新
46          pos = this.transform.position;
47          vel = this.rb.velocity;
48
49          Rebound ();              //境界判定
50          LimitVelocity ();        //速度制限
51          setGravity();            //重力の設定
52
53          manageWall(); //壁との衝突管理
54
55      }
56
57      //壁に衝突後、blindnessをtrueとし、
58      //一定時間経過後、blinednessをfalseに戻す。
        1 個の参照
59      private void manageWall(){
60
61          timeW += Time.deltaTime;
62          if(timeW < blindtime_after_wallhit){
63              blindness = true;
64          }else{
65              blindness = false;
66          }
67
68      }
```

manageWall()

SingleBoid.cs

SetVelocity(v)

```
158     //速度の設定
        0 個の参照
159     public void SetVelocity(Vector3 v){
160
161         //blindnessがtrueでなければ速度を更新する。 (2024.6.24)
162         if(!this.blindness){
163             this.rb.velocity = v;
164             this.vel = this.rb.velocity;
165         }
166
167     }
```

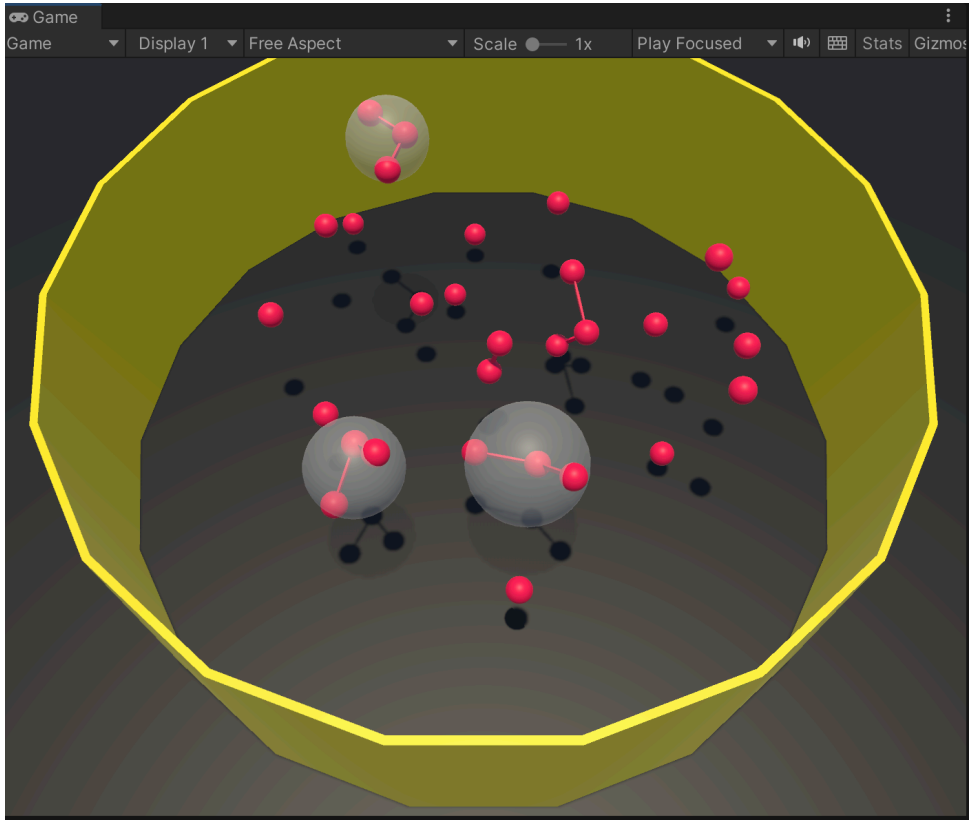
onCollisionEnter()

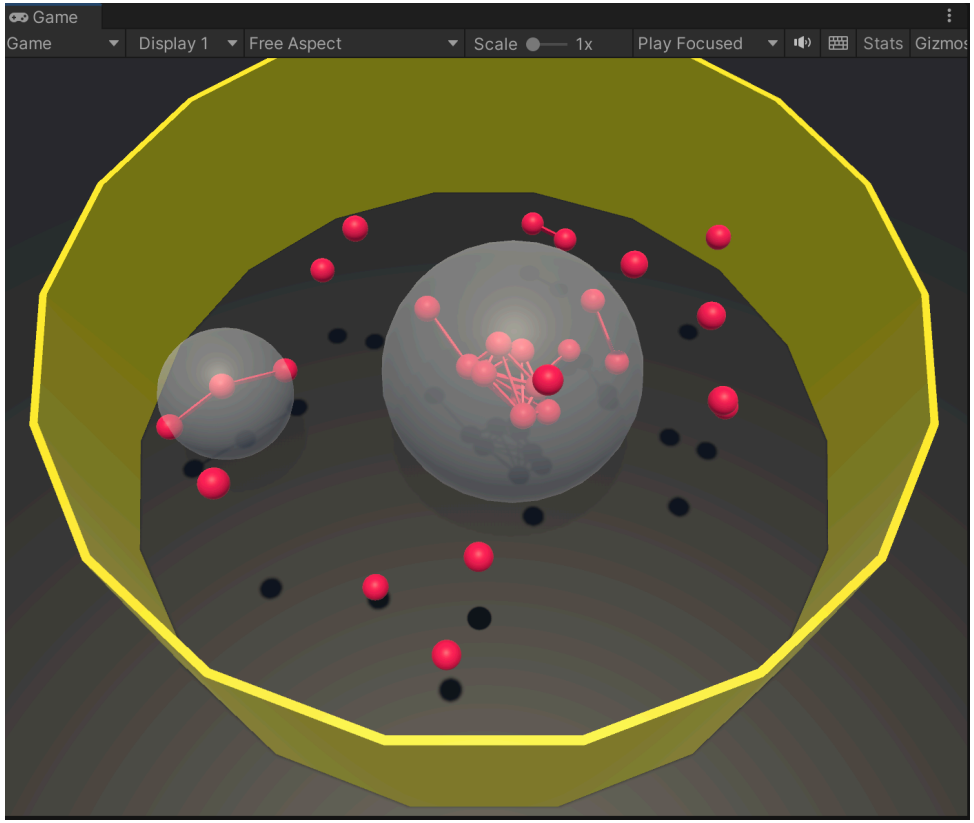
```
181     //壁への衝突を検知すると、timeWを0に更新。
        0 個の参照
182     void OnCollisionEnter(Collision hit)
183     {
184         if(hit.gameObject.tag == "Wall")
185         {
186             timeW = 0;
187         }
188     }
```

演習２- C1

集合知の解析

ボイドのパラメータを任意に変更し，
集団の振舞の変化を観察してください．





解析に関わる文（1/2）

BoidManager.cs

宣言部

```
//解析オブジェクト
BoidClusterAnalysis ana;
```

```
/* ボイド解析用の変数*/
private float countmax = 3000f; //計測フレーム

private float cls_sum = 0f;       //クラスターの総和
public float cls_count = 0f;      //現在のフレーム数
public float cls_mean = 0f;       //クラスターの平均値

//新しい知り合いできたフレーム数
//（10フレーム以上同一のクラスタにいるペアが新たに生まれた場合）
public int make_friends_frame = 0;
```

1000から、より大きな数字（3000～5000程度）に修正してください。

privateをpublicとしてください。

Start()

```
/* 解析オブジェクトの生成 */
ana = this.GetComponent<BoidClusterAnalysis> ();
```

解析の本体は、BoidClusterAnalysis.csに記述されています.

Update()

```
if (Input.GetKeyDown (KeyCode.I)) {
    InitBoidPosition ();
    InitBoidVelocity();

    cls_sum = 0f;
    cls_count = 0f;
    make_friends_frame = 0;
    ana.InitFriendsHistory(this);
}
```

Iボタンを押すと，計算用変数（総和とカウンタ）を0に初期化します.

解析に関わる文（2/2）

BoidManager.cs

Update()

```
/* 解析モードがON（Key A）のとき */
if (mode_analysis)
{
    ana.SetBoid(this);
    //個体数（1/10）をクラスター成立の要件とする
    ana.SetMinimumPop((int)Mathf.Floor(bsum / 10f));
    //クラスタを計算
    ana.CountCluster();
    //新しく「知り合い」が成立したボイドの数
    int new_friends = ana.UpdFriendsHistory(30);
    //クラスタに半透明の球を描画
    ShowClusterBall();

    //カウンタが最大値（初期値1000）となるまで、解析を続け
    if (cls_count < countmax)
    {
        //クラスタのフレーム内総数
        cls_sum += ana.csum;
        //クラスタの1フレーム平均
        cls_mean = cls_sum / cls_count;
        //計算フレーム数
        cls_count++;

        //新しい知り合いが一件でも成立すれば、
        //「知り合い成立機会」を一つ増やす
        if (new_friends >= 1)
        {
            make_friends_frame += 1;
        }
    }
}
```

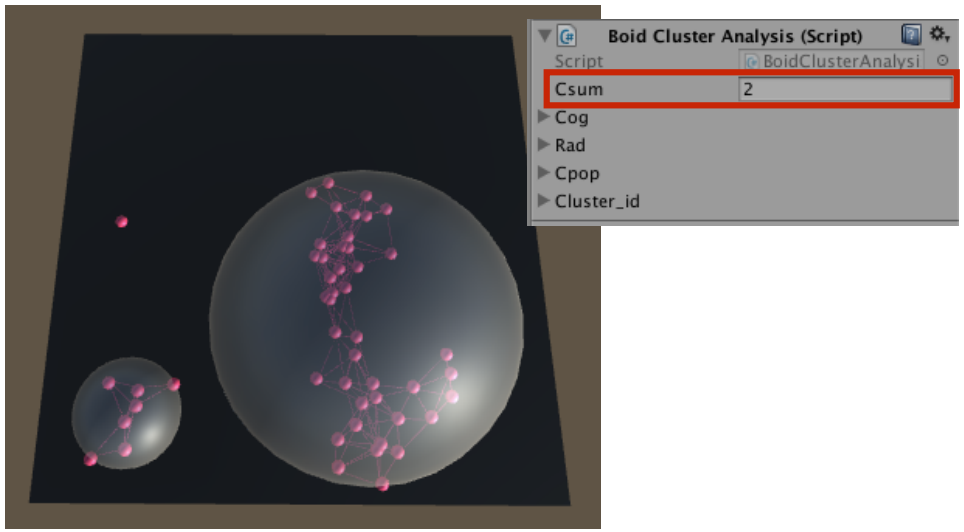
引数を「30」に修正してください。（引数は「知り合い」とみなす、連続クラスタ同居のフレーム数です）

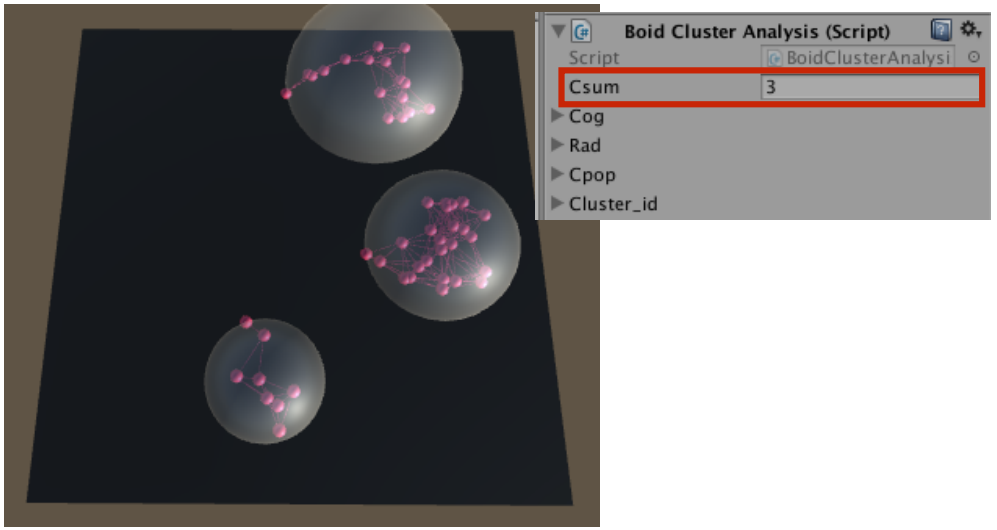
Aボタンで mode_analysis が true となると，クラスタのフレーム内平均（cls_mean）・知り合い成立回数（make_friends_frame）をフレーム数：countmax を上限として計算します．

A 解析モードの ON ／ OFF の切り替え

解析モードがONとなるとクラスターが可視化され, Boid Cluster Analysis コンポーネントの＜Csum＞にクラスター数が表示されます.

（ここでは, クラスターを＜個体数の1/10の可視距離内集団＞と定義しています）

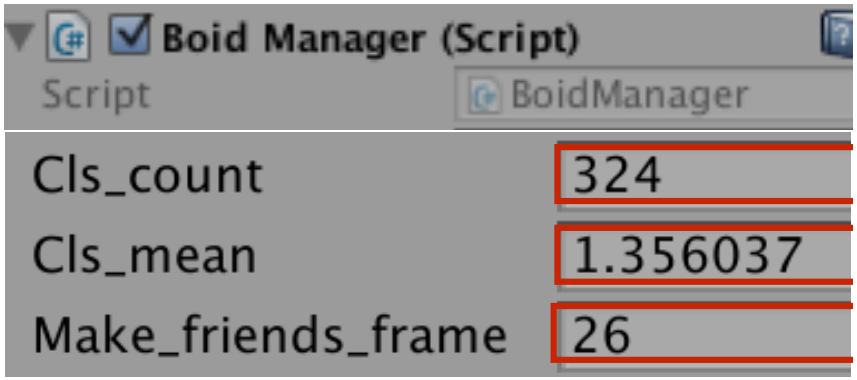




I ボイドの位置・速度の初期化

I ボタンを押した時点から, クラスターのフレーム内平均の計算・知り合い成立機会フレーム数のモニタが開始します.
Boid Manager コンポーネントに計算結果がリアルタイムで表示されます.

30フレームの間、同一のクラスタにいるボイドのペアについては「知り合い成立」とみなします.

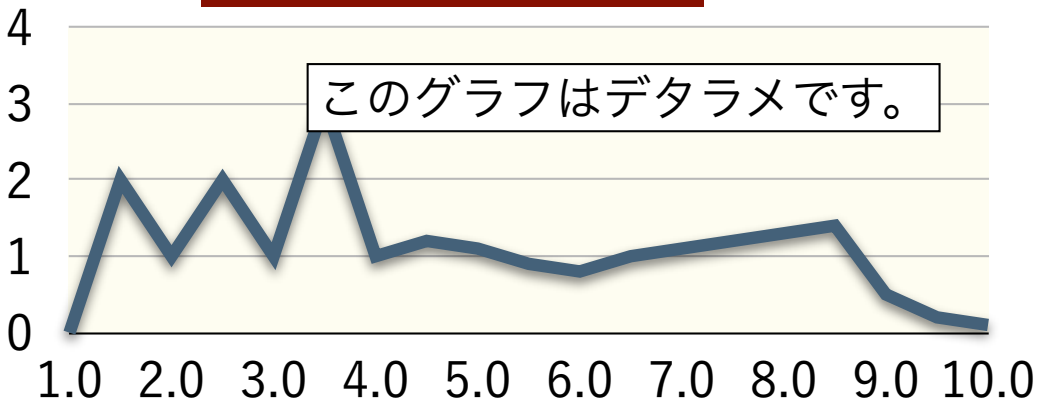


cls_count	計算フレーム数（最大10000）
cls_mean	クラスターのフレーム内平均
make_friends_frame	知り合い成立機会フレームの数

共通課題

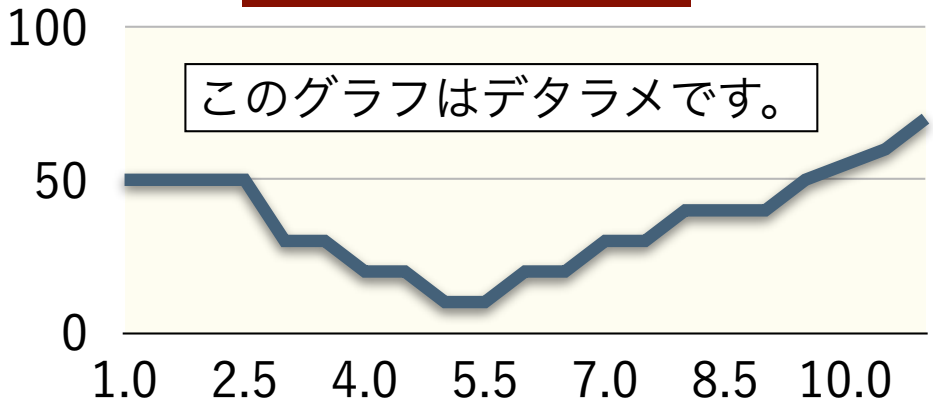
３次元空間における、ボイドの視界範囲とクラスタの数（フレーム内平均）および、知り合い成立機会（フレーム内総和）の関係を調べて、以下のようなグラフを作成してください。

クラスタ数（平均）



ボイドの視界範囲[m]

知り合い成立機会



ボイドの視界範囲[m]

ただし、視界範囲以外の変数は以下の値に固定してください。

```
neighbor_space = 1.0, pop = 30, c1 = 0.1; c2 = 5.0; c3 = 0.01;
```

上記の条件でグラフを作成し, グラフから読み取れることを考察してください. そのなかで, 自分が, この群衆のメンバーであることを想像して, 「**豊かな関係性とは何か**」に着目して考察してください.

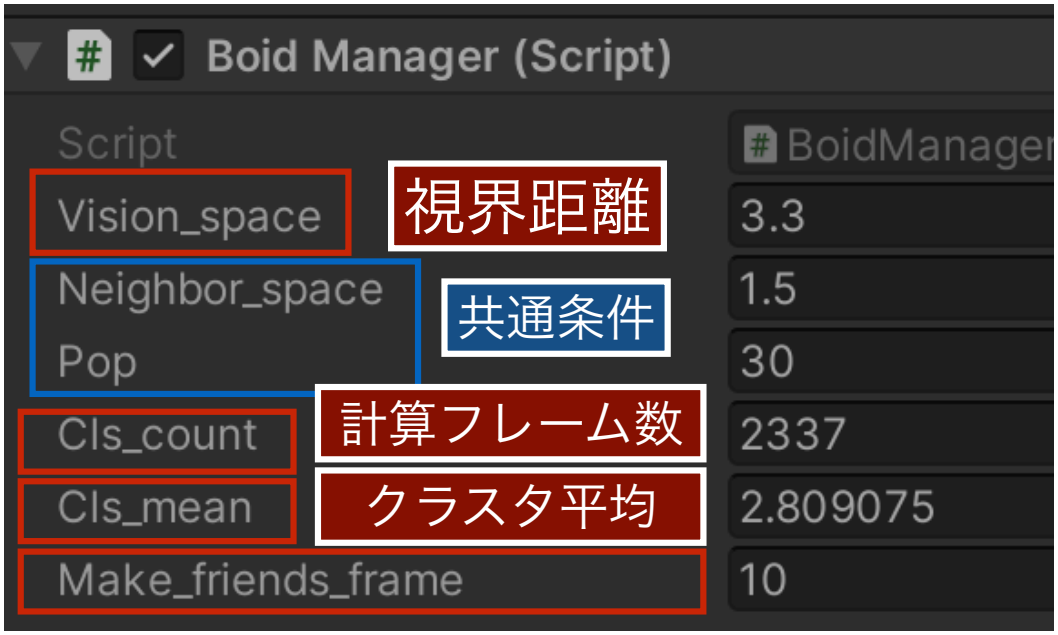
共通課題の詳細

3	三次元モードにします.
A	解析モードにします.
I	位置と解析を初期化します.

解析は、「Aキー」を押した後、「Iキー」を押してから約3000フレーム（計算機の特性に合わせて適当なところで切ってもらってOK）におけるフレーム内平均値（クラスタ）およびフレーム内総和（知り合い成立機会）として算出します.

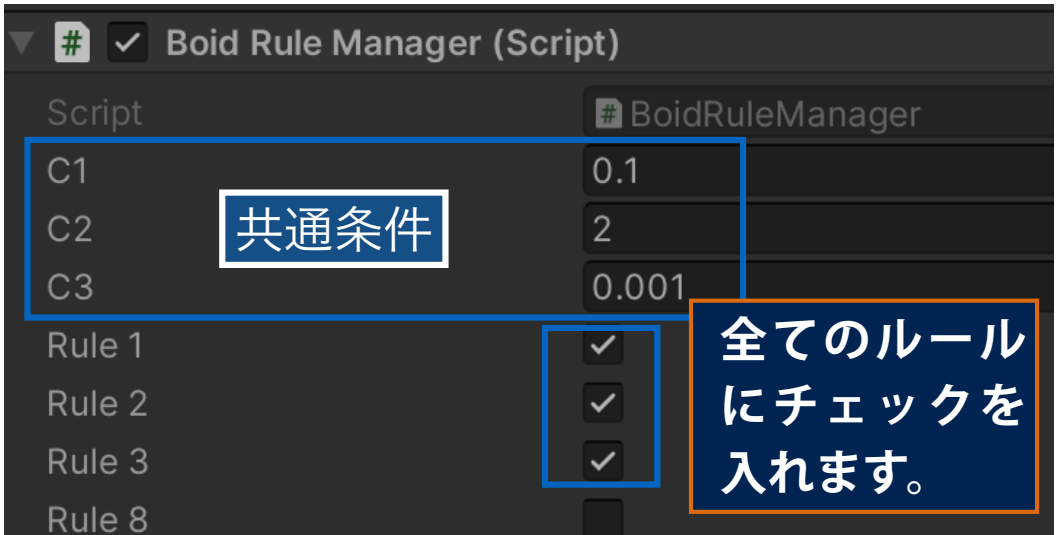
全ての視界距離の値を調べる必要はありません. グラフの概形がわかるために必要なデータ量を各自で判断してください. 同じ視界距離の値で複数回実行し、平均値を取るようにすると、より信頼性のある解析となります。

視界距離値は, 直接にGUIで vision_space の値を設定してください.



「I キー」を押した後, 3000フレームが経過すると, 自動的に計算が終了します.

知り合い成立機会



全てのルールにチェックを入れます。

自由課題

三次元空間ボイドにおいて、ルール１・ルール２・ルール３、または個体数（共通課題は30）がクラスタ形成あるいは「豊かな関係性の構築」にどのように関わっているかについて考察してください。

目視による考察でも良いが, データに基づいた考察が望ましい. この際、目的に応じて、共通課題とは異なる変数の値を選び、c1・c2・c3・popの値を変化させた時の「クラスタ数」「知り合い成立機会」の変化を計測するなどしてみてください。

考察の中で、自分が最も「豊かな関係性」と感じる「c1・c2・c3・pop」の変数の値を示してください。

意欲があれば、「BoidClusterAnalysis.cs」に独自のメソッドを追記し、新たな指標を導入してもよい。

課題C1（提出方法）

提出ファイル

共通課題と自由課題に関する書類を一つのフォルダに入れて, そのフォルダごと（「2250XX_WorkC1」）を提出してください.

提出先

https://lab.kenrikodaka.com/univclass/mediapractice2024/

Dropbox のファイルリクエストにより提出します. 投稿先のリンクは, 以下のページから.

期限

6/17（月）

（演習２）集合知と感染モデル				
05/15 [WEB資料]	UNITY環境のセットアップ	[資料PDF]	[YOUTUBE]	[出席報告]（小課題） !!5月20日締切
05/22	BOID（ルール１）	[資料PDF]	[YOUTUBE]	[出席報告]（課題） !!5月27日締切
05/31	BOID（ルール２・ルール３）	[資料PDF]	[YOUTUBE]	[出席報告]（課題） !!6月3日締切
06/05 – 06/12	集合知解析	[資料PDF]	[YOUTUBE]	[出席報告]は無し [課題C1提出]（集合知解析課題） !! 締切6月17日（月）