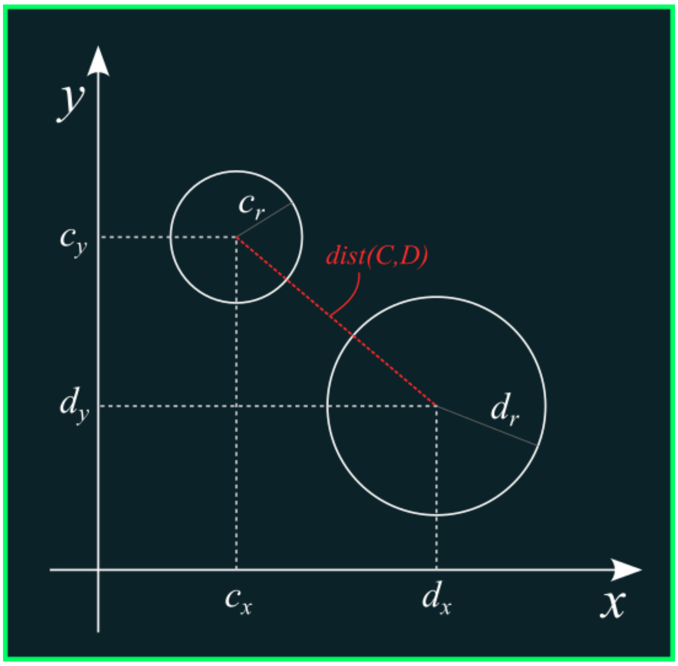
?????

Unityのコライダーのおかげで、当たり判定は自動的に計算され、「`OnCollisionXXX`」「`OnTriggerXXX`」で衝突があった時にお知らせを受け取る、処理ができるが、実際に裏側でなにを行っているのか理解してみましょう。

????

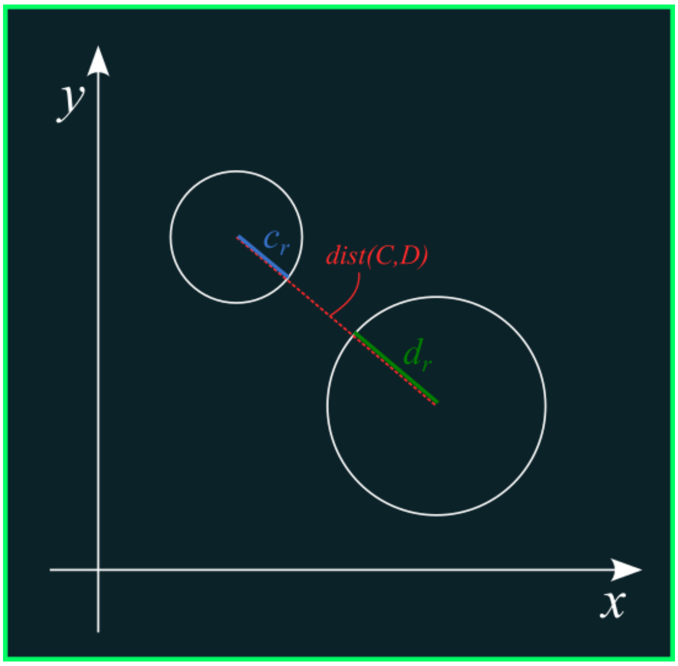
まず、一番楽なのは、球体同士の当たり判定である。以下の図を確認し、考えてみましょう

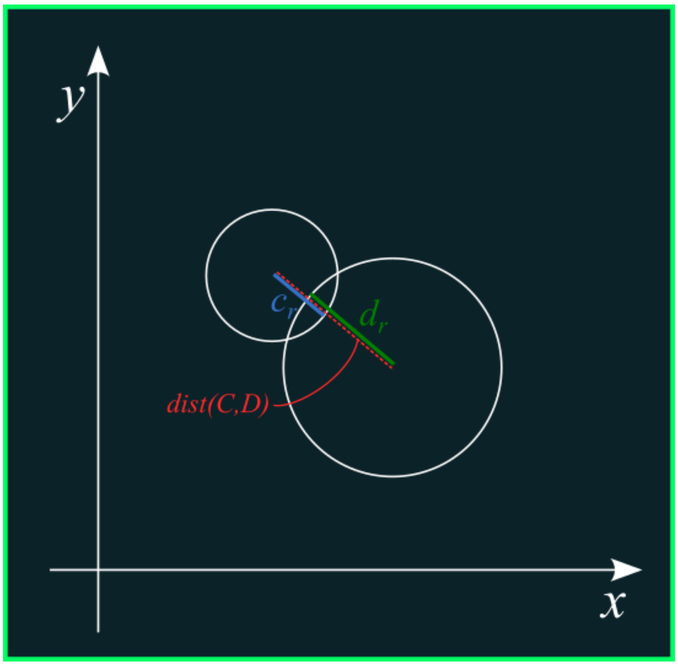


円「 **C**」（中心点 **(c_x, c_y)**、半径 **c_r** ）と「 **D**」（中心点 **(d_x, d_y)**、半径 **d_r** ）があるとして、中心の間の距離は、中心の差のノルムである：

$$dist(C, D) = \|(c_x, c_y) - (d_x, d_y)\|$$

この距離は半径の和よりも小さければ、２つの円が衝突しているといえる。





つまり

$$\begin{cases} \text{衝突する} & dist(C,D) \le (c_r + d_r)\text{の場合} \\ \text{衝突しない} & \text{それ以外} \end{cases}$$

??

Unityで コライダーなし の2つの球体を作成し、 MySphereCollider を作成し、半径を設定できるようにしてください（注意：コライダーの半径とメッシュの半径を合わせてください）

つぎ、 SphereCollisionCheck のスクリプトを作成し

- すべての MySphereCollider を取得
- コライダー同士の距離を計算
- 半径よりも小さければ、色を変える

MySphereCollider スクリプト

```
// 当たり判定：球体のコライダー
public class MySphereCollider : MonoBehaviour
{
□//半径
□[SerializeField]
□private float radius;
```

```
□//半径を返す
public float GetRadius()
□{
□□return radius;
□}

□//色を変える
public void ChangeColor(Color color)
□{
□□// MeshRenderer□□を変える
MeshRenderer mr = GetComponent<MeshRenderer>();
□□mr.material.color = color;
□}
}
```

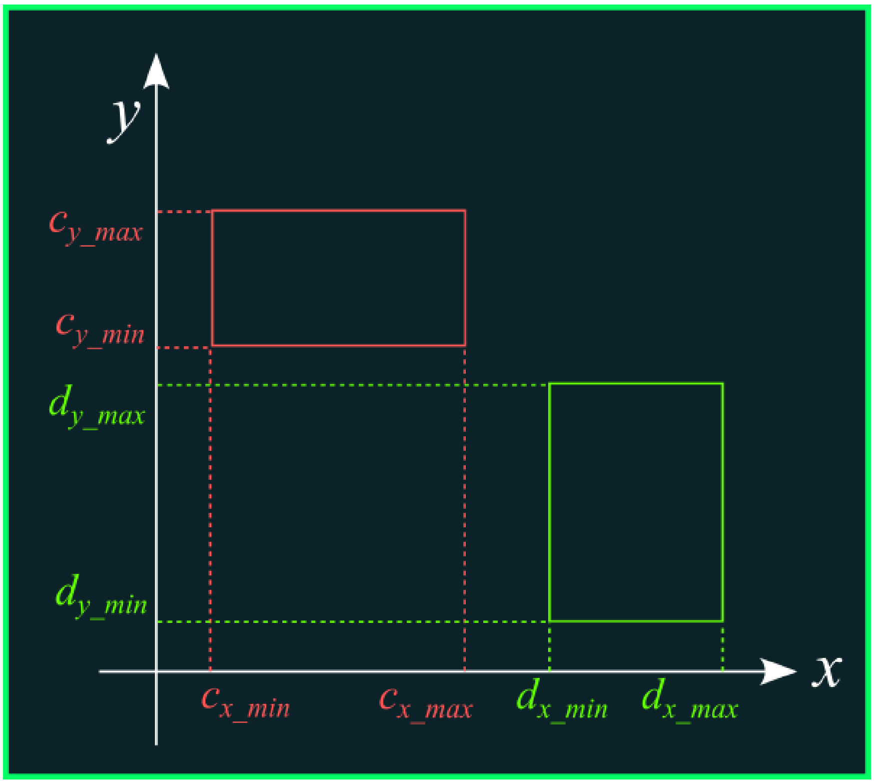
SphereCollisionCheck スクリプト

```
// コライダー同士の当たり判定を確認
public class SphereCollisionCheck : MonoBehaviour
{
□//すべてのコライダー
private MySphereCollider[] colliders;

□private void Start()
□{
□□// FindObjectsByType 「このスクリプトを付いているオブジェクトのすべて下さい」
colliders = FindObjectsByType<MySphereCollider>(FindObjectsSortMode.None);
□}

□void Update()
□{
□□/□期値はすべて「false」
bool[] isCollision = new bool[colliders.Length];

□□/□部 vs 全部を比較
for (int i = 0; i < colliders.Length - 1; i++)
```

```
    {
        for (int j = i + 1; j < colliders.Length; j++)
        {
            MySphereCollider iObje = colliders[i];
            MySphereCollider jObje = colliders[j];

            //
            float distance = (iObje.transform.position - jObje.transform.position).magnitude;

            //
            float radiusSum = iObje.GetRadius() + jObje.GetRadius();

            //
            if (distance - radiusSum <= 0)
            {
                isCollision[i] = true;
                isCollision[j] = true;
            }
        }
    }

    //
    for (int i = 0; i < isCollision.Length; i++)
    {
        if (isCollision[i])
            colliders[i].ChangeColor(Color.red);
        else
            colliders[i].ChangeColor(Color.white);
    }
}
```

?????????

BoxCollider C D



この場合は:

- ある箱の「最大X」は他の箱の「最小X」よりも小さければ → 衝突しない
- ある箱の「最大Y」は他の箱の「最小Y」よりも小さければ → 衝突しない
- それ以外、衝突する

3次元の場合はZも含める。

??

Unityでコライダーなしの2つの立方体を作成し、 MyBoxCollider を作成し、幅、高さ、深さを設定できるようにしてください(注意: コライダーの半径とメッシュの半径を合わせてください)

つぎ、 BoxCollisionCheck のスクリプトを作成し

- すべての MyBoxCollider を取得
- 各軸の「最大」と「最小」を比較
- 衝突したら、色を変える

MyBoxCollider スクリプト

```
// 箱コライダー
public class MyBoxCollider : MonoBehaviour
{
```

```
    // サイズ（幅、高さ、深さ）
    [SerializeField]
    private Vector3 size;

    // 各軸の最大位置を返す
    public Vector3 GetMax()
    {
        return transform.position + size / 2;
    }

    // 各軸の最小位置を返す
    public Vector3 GetMin()
    {
        return transform.position - size / 2;
    }

    // 色を変える
    public void ChangeColor(Color color)
    {
        // MeshRendererの色を変える
        MeshRenderer mr = GetComponent<MeshRenderer>();
        mr.material.color = color;
    }
}
```

BoxCollisionCheck スクリプト

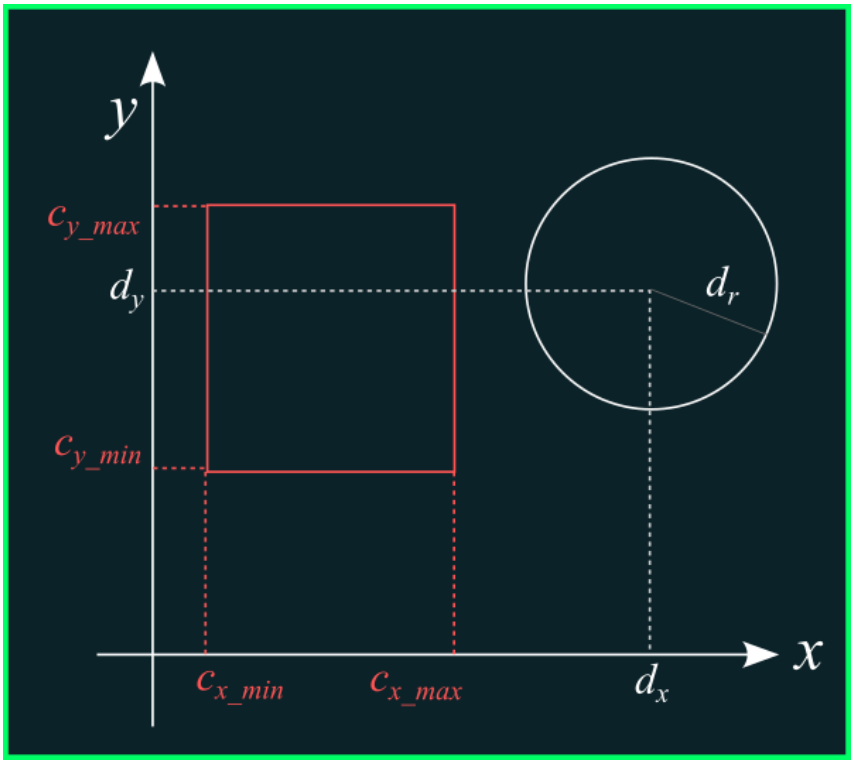
```
// MyBoxCollider同士の当たり判定を確認
public class BoxCollisionCheck : MonoBehaviour
{
    // すべてのコライダー
    private MyBoxCollider[] colliders;

    private void Start()
    {
        // FindObjectsByType = 「このスクリプトを付いているオブジェクトのすべて下さい」
        colliders = FindObjectsByType<MyBoxCollider>(FindObjectsSortMode.None);
```

```
    }

    void Update()
    {
        // 初期値はすべて「false」
        bool[] isCollision = new bool[colliders.Length];

        // 全部 vs 全部を比較
        for (int i = 0; i < colliders.Length - 1; i++)
        {
            for (int j = i + 1; j < colliders.Length; j++)
            {
                // 最大と最小位置を取得
                Vector3 iMax = colliders[i].GetMax();
                Vector3 iMin = colliders[i].GetMin();
                Vector3 jMax = colliders[j].GetMax();
                Vector3 jMin = colliders[j].GetMin();

                // 軸ずつを確認
                if (iMax.x < jMin.x) continue;
                if (iMax.y < jMin.y) continue;
                if (iMax.z < jMin.z) continue;

                // 逆の比較も
                if (jMax.x < iMin.x) continue;
                if (jMax.y < iMin.y) continue;
                if (jMax.z < iMin.z) continue;

                // 当たった!
                isCollision[i] = true;
                isCollision[j] = true;
            }
        }

        // 衝突しているコライダーだけの色を変える
        for (int i = 0; i < isCollision.Length; i++)
        {
            if (isCollision[i])
```

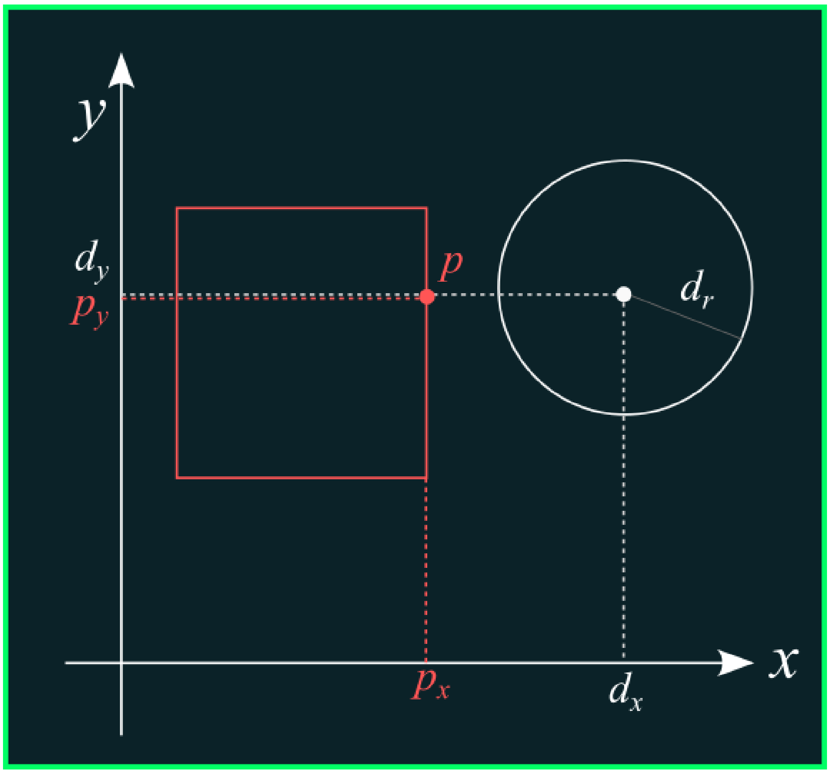
```
                colliders[i].ChangeColor(Color.red);
            else
                colliders[i].ChangeColor(Color.white);
        }
    }
}
```

????

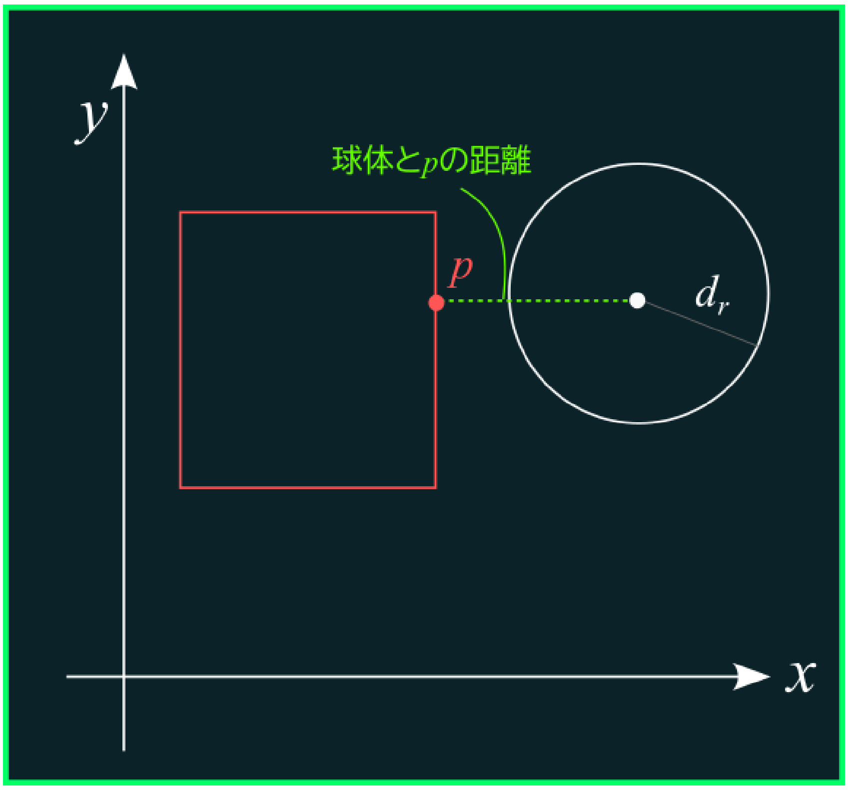
SphereCollider と　BoxCollider の当たり判定もあるので、その処理を学ぼう。



ここで、球体「　　　D」の中心点　　（ d_x, d_y）に、箱のある最も近い点「　　　　$p=(p_x, p_y)$」を求める:



そして、球体同士と同じように、球体の中心と点「p」の距離が半径よりも小さければ、衝突になる。



??

BoxSphereCollisionCheck を作成し、

- MyBoxCollider と MySphereCollider のすべてを求める
- １個ずつの球体に対して、すべての箱を確認
- 球体に一番近い点を求めて、半径よりも小さければ、衝突にする

BoxSphereCollisionCheck の当たり判定

```
// 球体と箱の当たり判定
public class BoxSphereCollisionCheck : MonoBehaviour
{
    // すべての箱
    private MyBoxCollider[] boxes;

    // すべての球体
    private MySphereCollider[] spheres;

    void Start()
    {
        boxes = FindObjectsByType<MyBoxCollider>(FindObjectsSortMode.None);
        spheres = FindObjectsByType<MySphereCollider>(FindObjectsSortMode.None);
    }

    void Update()
    {
        // 初期値はすべて「false」
        bool[] isBoxCollision = new bool[boxes.Length];
        bool[] isSphereCollision = new bool[spheres.Length];

        // すべての球体 vs すべての箱
        for (int i = 0; i < spheres.Length; i++)
        {
            for (int j = 0; j < boxes.Length; j++)
            {
                // コライダーを準備
                MySphereCollider s = spheres[i];
                MyBoxCollider b = boxes[j];

                // 球体の中心
                Vector3 center = s.transform.position;

                // 箱の最大と最小位置
                Vector3 max = b.GetMax();
                Vector3 min = b.GetMin();
```

```
                // 球体の中心に一番近い点「p」を求める
                Vector3 p = new Vector3();
                p.x = Mathf.Clamp(center.x, min.x, max.x);
                p.y = Mathf.Clamp(center.y, min.y, max.y);
                p.z = Mathf.Clamp(center.z, min.z, max.z);

                // 中心からpまでの距離が小さければ、当たっている
                float dist = (p - center).magnitude;
                if (dist < s.GetRadius())
                {
                    isSphereCollision[i] = true;
                    isBoxCollision[j] = true;
                }
            }
        }

        // 衝突しているコライダーだけの色を変える
        for (int i = 0; i < isBoxCollision.Length; i++)
        {
            if (isBoxCollision[i])
                boxes[i].ChangeColor(Color.red);
            else
                boxes[i].ChangeColor(Color.white);
        }
        for (int i = 0; i < isSphereCollision.Length; i++)
        {
            if (isSphereCollision[i])
                spheres[i].ChangeColor(Color.red);
            else
                spheres[i].ChangeColor(Color.white);
        }
    }
}
```

Revision #2
Created 2026-06-24 01:26:36 UTC by Menendez Francisco
Updated 2026-06-24 04:18:47 UTC by Menendez Francisco