





```
Colisões

public class PShape05 : MonoBehaviour {
    [HideInInspector]
    public Vector3 position;
    public float radiuns = 0.5f;
    public bool isTrigger;

    void Start() {
        position = transform.position;
    }
}
```

```
void Update() {
    iff (lip*layer){
        Vector3 dir = new Vector3();
        dir.x = Input.GetAxie("Horizontal");
        dir.y = Input.GetAxie("Horizontal");
        dir.y = Input.GetAxie("Horizontal");
        shape.position += dir.* * speed * Time.deltaTime;
    }
Colisões
public class PBody05 : MonoBehaviour {
[HideInInspector] public PShape05 shape;
public float mass = 1.0f;
                                                                                                                     }
}
void FixedUpdate() {
DynamicMotion();
KinematicMotion();
CheckCollision();
if (lisStatic) transform.position = shape.position;
public bool isStatic;
 public bool useGravity;
public bool isKinematic
[Header("Info")]
public Vector3 velocity:
public Vector3 venderly,
public Vector3 acceleration;
public Vector3 gravity;
public Vector3 force;
                                                                                                                     }
void DynamicMotion() {
    acceleration = force / mass + gravity;
    velocity += acceleration * Time.fixedDeltaTime;
}
 public bool isPlayer;
                                                                                                                        }
void KinematicMotion() {
    shape.position += velocity * Time.fixedDeltaTime;
 void Start() {
if (useGravit
    if (useGravity) gravity=new Vector3(0,-9.81f,0);
shape = GetComponent<PShape05>();
                                                                                                                        void CheckCollision() {
    PController05.instance.CheckCollision(this);
```

```
Complexidade (0²)

• Deve-se testar cada objeto contra todas os outros
Otimização

• Determinar objetos com chance de colisão (ex.: estáticos não se movem)

• Divisão de espaço (Árvore Binária, QuadTree, OctTree)

• Utilizar calculo de colisão mais simples quando a distância é maior.
```

Colisões Duas etapas: Broad phase: • Fase que detecta potencial colisão. • Por questões de desempenho são usadas tecnicas como: • Space partition • Uniform Grid • Quad trees (2D) • Cotree (3D) • Sort and Sweep Algorithm (Bullet em btAxisSweep3) • Bounding volume. Narrow Phase: • fase que detecta colisão de fato. • Maior precisão nos cálculos. Detectar ponto exato da colisão.

```
Colisões

void Start() {
    instance = this;
    GameObject[list = GameObject.FindGameObjectsWithTag("Body");
    Debug Log(list Length);
    bodys = new PBodyOS[list.Length];
    for (int i = 0; i < list.Length; i++){
        bodys[] = isst[].GetComponent<PBodyOS>();
    }
} void ResolvePenetration (PCollision05 c, PBodyO5 b1, PBodyO5 b2){
    if (b2.lisStatic){
        b1.shape.position += c.penetration *-c.normal;
    } else {
        b1.shape.position += c.penetration * 0.5f *-c.normal;
        b2.shape.position += c.penetration * 0.5f * c.normal;
    }
}
```

```
Engines de Física
Rigidbody x Objeto (Collider não Trigger)
                                             OnCollision
                                                                 Para ao colidir
                                                  OK
                           velocity
                                                                   (aSleep)
     Rigidbody
                        Translate() ou
                                                              Não atravessa mas
                           position
                                                                  fica batendo
                                                                  Não se move
                           velocity
                                                 Não
     Rigidbody
                        Translate() ou
    isKinematic
                                                 Não
                                                               Atravessa o objeto
                           position
    Objeto sem
Rigidbody
                        Translate() ou
                                                                  Atravessa o
                                                 Não
                           position
                                                                   Rigidbody
```

Engines de Física Rigidbody x Objeto (Collider Trigger)		
Script	OnTrigger	Visualmente
velocity	OK	Atravessa o objeto
Rigidbody Translate() ou position	OK	Atravessa o objeto
velocity	OK	Não se move
Rigidbody isKinematic Translate() ou position	OK	Atravessa o objeto
Translate() ou position	Não	Atravessa o Rigidbody
	Script velocity Translate() ou position velocity Translate() ou position Translate() ou position	Collider Trigger) Script OnTrigger velocity OK Translate() ou position OK velocity OK Translate() ou OK Translate() ou OK Translate() ou OK

