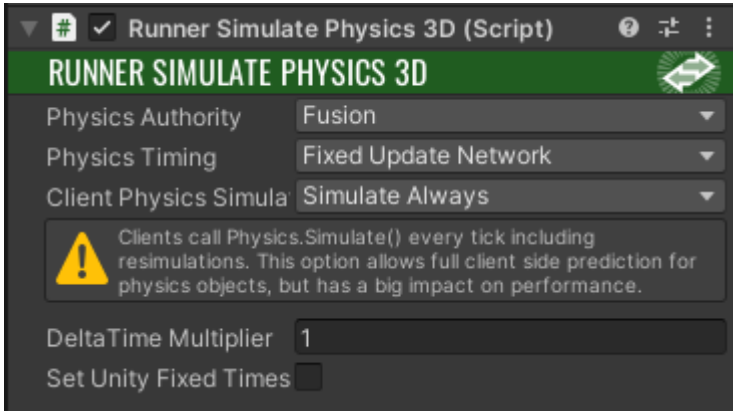
Physics trong Photon Fusion

nguyenkimlong@savameta.com

Cài đặt

Để physics có thể hoạt động với Fusion, ta phải thêm component **RunnerSimulatePhysics3D** vào GameObject chứa NetworkRunner

Sau khi thử nghiệm các cách khác nhau, thì cài đặt như hình bên giúp di chuyển mượt nhất đối với vật lý cầm nắm cho bàn tay



Runner.SetIsSimulated(Object, true);

Trong bản cập nhật của Fusion 2, tính năng chạy **FixedUpdateNetwork()** trên **proxy** không còn là mặc định

Để bật nó lên ta phải dùng hàm **SetIsSimulated**:

- Object: là NetworkObject cần tính năng này
- true/false: là việc bật/tắt tính năng này

Đối với vật lý cầm nắm cho Fusion thì bật tính năng này lên là cần thiết

*__proxy__: là những người chơi khác trên máy của mình

Utilities

Để tính vận tốc, ta lấy **pos2** trừ **pos1** rồi chia cho **dt**

Cách tính vận tốc góc thì hơi khác một chút!

```
public static void SetVelocity(this Rigidbody rb, Vector3 pos1, Quaternion rot1, Vector3 pos2,
Quaternion rot2, float dt)
{
    rb.velocity = (pos2 - pos1) / dt;
    rb.angularVelocity = GetAngularVelocity(rot1, rot2, dt);
}
```

Utilities

Ta cần quy đổi ra **angle** và **axis** rồi mới chia cho **deltaTime**

* Việc quy đổi là cần thiết vì **Rigidbody.angularVelocity** chỉ nhận **Vector3**

```
public static Vector3 GetAngularVelocity(Quaternion current, Quaternion target, float dt)
{
    (target * Quaternion.Inverse(current)).ToAngleAxis(out var angle, out var axis);

    return (angle > 180 ? angle - 360 : angle) * Mathf.Deg2Rad * axis / dt;
}
```

RenderTimeframe

Tính năng này cho phép thay đổi **Tick** giữa **Local** và **Remote**

Cách áp dụng:

- Nếu vật đang được cầm bởi **Player**: Đổi sang **Tick Local**
- Nếu vật đang được cầm bởi **Proxy**: Đổi sang **Tick Remote**

*Đây chính là nguyên nhân khiến vật bị lag sau khi ném

```
Object.RenderTimeframe = hasInput ? RenderTimeframe.Local : RenderTimeframe.Remote;
```

Data

```
public struct InputData : INetworkInput
{
    public Vector3 HeadPosition;
    public Quaternion HeadRotation;

    public Vector3 LeftHandPosition;
    public Quaternion LeftHandRotation;
    public GrabInfo LeftGrabInfo;

    public Vector3 RightHandPosition;
    public Quaternion RightHandRotation;
    public GrabInfo RightGrabInfo;

    public Vector2 MoveDirection;
    public Vector2 RotateDirection;
}
```

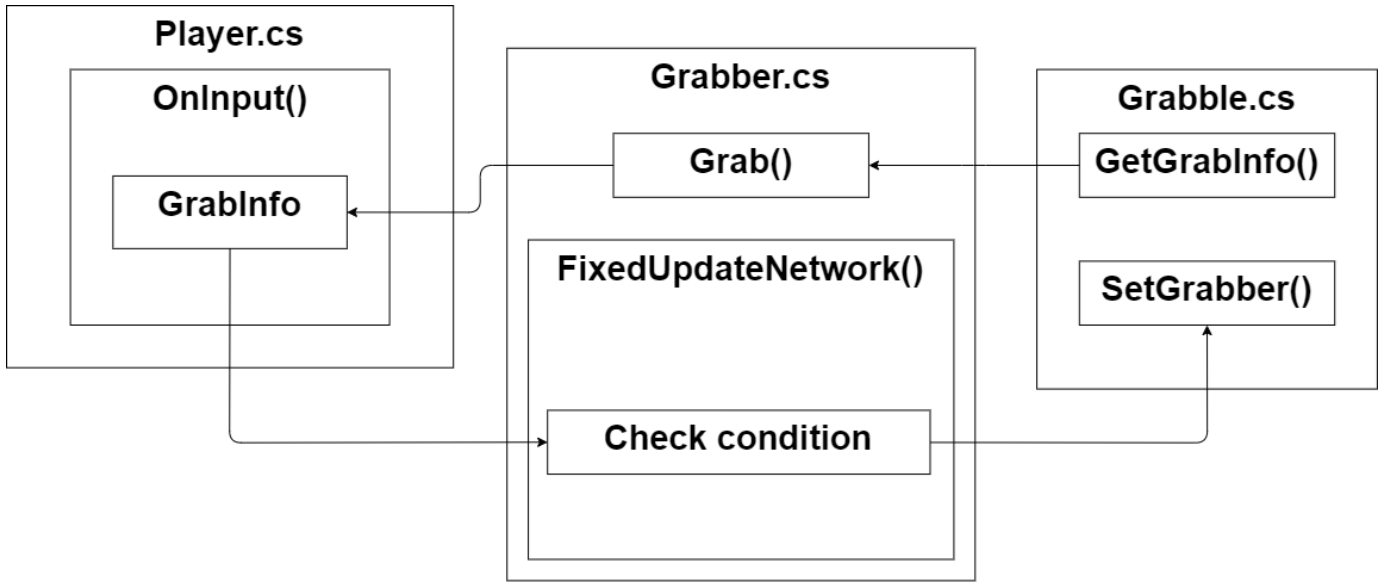
```
public enum GrabberSide
{
    Left,
    Right
}

public struct GrabInfo : INetworkStruct
{
    public GrabberSide GrabberSide;

    public NetworkBehaviourId GrabberId;
    public NetworkBehaviourId GrabbleId;

    public Vector3 PositionOffset;
    public Quaternion RotationOffset;
}
```

Implement


Player.cs
OnInput()
GrabInfo
Grabber.cs
Grab()
FixedUpdateNetwork()
Check condition
Grabble.cs
GetGrabInfo()
SetGrabber()

Demo