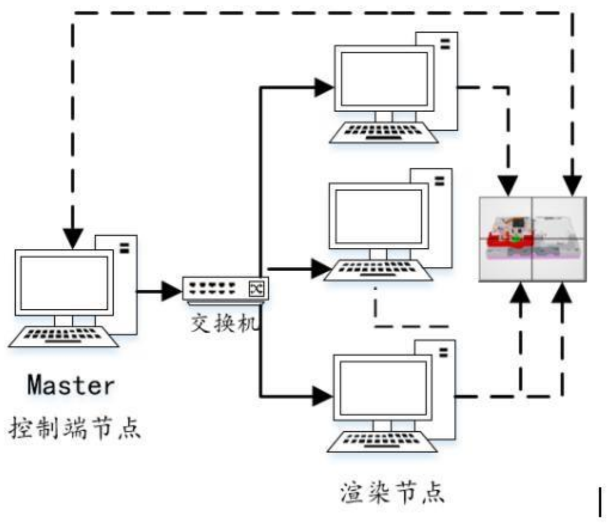
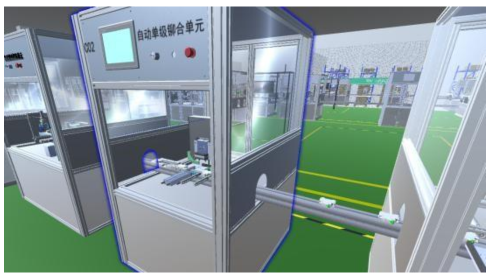
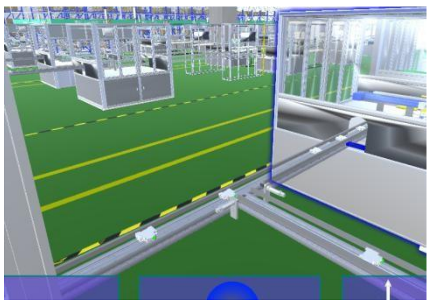
This paper describes the distributed rendering that the author has realized,The solution is communication-oriented, and the transmission of rendering pipeline data is not involved between the master and the slave. What is transmitted between master and slave is the position information between moving parts (circuit breakers) and the script of switching cameras between master and slave based on RPC,The specific implementation mechanism will be presented in the specific Unity code.



I still try to explain the working mechanism of this architecture. The working principle of the architecture is mainly in two aspects: one is the control of the power sources of the four host worlds, and the other is the switching of cameras among the four worlds.

power source

Introduction of power source

It is worth noting that in order to facilitate readers to understand the specific situation of a single project, I will provide a short video of 56 seconds. The power source of the project: the physical engine based on Unity is applied to the circuit breaker by the track, that is, the small square on the drawing, and the other is the cylinder motion based on 3dmax animation. The movement of the cylinder requires the breaker to move to a specific position where the trigger is placed. When the trigger is triggered, the animation of the cylinder will be triggered to change the position and angle of the circuit breaker. It is quite cumbersome to implement the whole trigger mechanism to the specific code, and I strive to give you an intuitive feeling. On the other hand, as long as the power of the track is cut off, the cylinder will not move,In other words, the power of the track is the power source of the whole project.

Power source cut-off mechanism

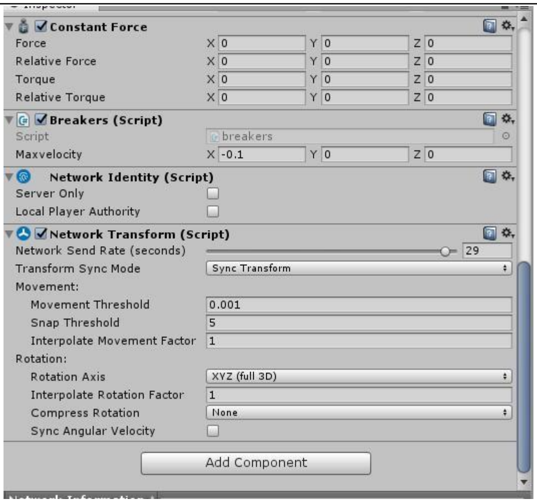
It is worth noting that at the beginning, the world scenes of the virtual workshops in the four hosts are exactly the same, which is the necessary prerequisite for realizing the synchronization of the four worlds.

I cut off all the orbital power sources in the four worlds before the four hosts established local area network connection,At this time, all the scenes in the four worlds are still. When the other three slave machines establish contact with the only master machine, all the power tracks in the world of the master machine are opened, and the power in the scenes of the other three slave machines remains cut off, which is a prerequisite for the master machine to synchronize the circuit breaker of the slave machine.

Camera operating mechanism.

As the reader can see, there are two cameras in the scene, one is the player camera with large field of vision and the other is the roaming camera with relatively small field of vision. The camera has the movement and rotation given by the script, which is controlled by the keyboard and mouse. Another roaming camera has its own fixed roaming route, and its motion is given by the established script, which is an automatic roaming. In addition, a script is set in the project,When the keyboard and mouse are not operated for a period of time, the camera will switch from the player camera in the big scene to the roaming camera,Once a user operates the mouse and keyboard during the roaming of the roaming camera, the scene will immediately switch from the roaming camera to the player camera in the big scene.

同步具体机制



```
Constant Force
Force                         X 0      Y 0      Z 0
Relative Force                X 0      Y 0      Z 0
Torque                        X 0      Y 0      Z 0
Relative Torque               X 0      Y 0      Z 0
Breakers (Script)
Script                        breakers
Maxvelocity                   X -0.1   Y 0      Z 0
Network Identity (Script)
Server Only                   ☐
Local Player Authority        ☐
Network Transform (Script)
Network Send Rate (seconds)   29
Transform Sync Mode           Sync Transform
Movement:
  Movement Threshold          0.001
  Snap Threshold              5
  Interpolate Movement Factor 1
Rotation:
  Rotation Axis               XYZ (full 3D)
  Interpolate Rotation Factor 1
  Compress Rotation           None
  Sync Angular Velocity       ☐
```

As mentioned above, the power source of the master terminal is turned on, but the power source of the slave terminal is still in a cut-off state, so how to synchronize the breaker movement of the master terminal to the slave terminal? I use networktransform, a network component of unity, which is a packaged component of Unity,If you want to know the specific information of this component, you can query my other reference materials or query the documents about Unity. This module will send the information such as the position and angle of the object added with this module to the other three slave terminals at a sending frequency of 29 times per second (this number can be adjusted to 30 at the maximum),When the slave terminals receive the information, they will synchronize the position of the object attached with this module. This component is added to all motion circuit breakers in this project, so that the circuit breaker at the slave end will update the position of its own circuit breaker in real time to achieve synchronization. It is worth noting that the trigger mechanism of the cylinder at the slave end has not been destroyed,When the circuit breaker at the slave end receives the information from the master end and moves to a specific position, the cylinder will still make a call animation in the world of the slave end,So far, the movement of the whole virtual workshop is ideally synchronized. The next step is the synchronization of camera switching, which is through RPC mechanism,When the camera detects that the master terminal is from the big scene camera to the roaming camera or from the roaming camera to the big scene camera, it will tell the slave terminal to call the local RPC function, so as to achieve the purpose of switching cameras.

```
[Command]// determine whether the camera at the main end has changed,public void
    cmddecadecameraischange () {if.  (freeCammIsOn     ! =   recordLastfreeCammIsOn)    {
        weatherChange=   true;
recordLastfreeCammIsOn = freeCammIsOn;  }
        else { weatherChange= false; }
    }
    [Command]// control the client to switch the camera on and off.
```

```
        public void CmdChangeCam()
               { RpcChangeCam();
        }
        [ClientRpc] public void
        RpcChangeCam()
        {
               if (isServer) { return;  } freeCam.SetActive(!
               (freeCam.activeInHierarchy));  autoCam.SetActive(!
               (autoCam.activeInHierarchy));
        }
        // Use this for initialization void Start ()
        { freeCammIsOn = freeCam.activeInHierarchy;
        recordLastfreeCammIsOn = freeCammIsOn;
        autoCamIsOn = autoCam.activeInHierarchy;

        // Update is called once per frame void Update ()
        { freeCammIsOn = freeCam.activeInHierarchy;
        CmdDecideCameraIsChange();  if
        (weatherChange) { CmdChangeCam(); }
        }
}
```

Assuming that this script is A.cs, it is worth noting that the slave must also have an A.cs, but the slave's A.cs only keeps the above.
RpcChangeCam function, because the slave can execute this function as long as it receives the request of the master. Other functions of are redundant to the slave side.

And variable interpretation.

freeCammIsOn	Bool type detects whether the camera in large scene is hidden in the scene in real time.
recordLastfreeCammIsOn	Record the bool value before freeCammIsOn executes the assignment statement in real time.
weatherChange	Judging whether the camera in large scene has changed its existing state includes:
	From hidden to rendered and from rendered to hidden.
CmdDecideCameraIsChange	This code can only be executed in the main terminal. This function is to assign a value to the weatherChange

	variable, that is, to determine whether the state of the camera in the big scene has changed.
RpcChangeCam	If weatherChange is true, it will tell the slave that it needs to switch cameras, and the slave will switch cameras through this function.

In principle, the cameras of the four worlds here are completed synchronously.

Project experiment questions:

I recorded more than 4 minutes of videos, so please watch them patiently.

1. Using RPC to switch cameras from end to end has encountered considerable delay, which leads to the dislocation of four screens.
2. The jitter visible to the naked eye will occur at the slave end of the circuit breaker, which should be caused by the position signal transmitted by the network.

When the camera switching happens at the master end, the master end sends an instruction to switch cameras to the slave end, the signal is transmitted to the slave end, and then the slave end parses and calls the corresponding rpc function, which requires spending corresponding resources and time at each stage. This is also the reason for the switching delay of the slave camera. This delay seems to be technically difficult to achieve.

The reason of circuit breaker jitter may be related to the frequency of transmission, the delay of the same signal as above, and the underlying interpolation algorithm of the set price itself. I can't solve this jitter problem perfectly by setting the transmission frequency as the highest 30 .