============= =============

============= Unity Tips =============

============= =============

@ Canvas

Render Mode : 采用 Screen Space - Camera 模式

添加 Canvas Scaler 组件 用于自适应屏幕

UI Scale Mode : 采用 Scale With Screen Size 模式 大部分都选择该模式

Reference Resolution ： X , Y 分别为开发指定的大小 在缩放过程中则会以此为参考进行自适应

Screen Match Mode ：采用 Expand 模式

@ 一些不需要响应事件的obj, 例如 text 等, 则可以将 Raycast Target 的 √ 去掉 可以提高一点性能

@ Outline 组件

自带的 描边 功能 Effect Color ：描边颜色 Effect Distance ：X Y 方向描边大小 Use Graphic Alpha ：是否要受透明度的影响

@ Shadow 组件

自带的 阴影 功能 参数类似于 描边

\*\* 值得注意的是 使用这两个组件 会导致 顶点数翻倍

@ Toggle Group 与 Toggle 的配合

若 有 A 和 B 两个选项 并且两者互斥 只能选一个

那么 A B 挂上 Toggle 组件 A B 其中一个再挂上 Toggle Group组件

两者 Toggle 组件中的 Group 选择 挂了 Toggle Group组件的该 obj

Toggle 属性中的 Graphic ： 选中的状态图显示

Toggle Group 属性中的 Allow Switch Off ： 勾上则可以让所有的 toggle 状态都为 未选中 否则所有 toggle Group中的 toggle 必然只有一个为选中

@ Content Size Fitter 组件

Horizontal Fit 和 Vertical Fit 也就是 水平 和 垂直 方向上的调整

分别有 三个值 ：Unconstrained 不执行任何调整 MinSize ：调整内容的最小大小 PreferredSize ：调整内容的首选大小 会自动根据内容调整大小

需要调整的时候 一般都选为 PreferredSize 模式

@ Vertical Layout Group 垂直布局 Horizontal Layout Group 水平布局 Grid Layout Group 网格布局

配合 Scroll Rect 组件使用

@ Text 组件

Horizontal overflow 垂直溢出处理 选择 wrap 则会隐藏溢出部分 选择 overflow 则会保留

Vertical overflow 水平溢出处理 选择 truncate 则会截断溢出部分 选择 overflow 则会保留

可以通过这个属性来 调节 text 是否受到 Horizontal Fit 等这些组件的影响

@ 缩小 apk 体积

1. 图片大小改为 2 的 N 次方 , 最大为 1024\*1024

2. 图片属性设置可根据情况更改 maxSize , format , filter mode

3. 打包图集 window -- sprite packer 需要打包在一起的图片 packingtag 属性 改为统一的标志

4. 移除无用的资源 resources 文件夹中的 Hierarchy 视图中引用到的无用资源 prefab 中引用到的无用资源

5. 移除一些无用的脚本

@ unity 导入 spine 动画

3个文件 ---- >> json .atlas.txt .png

@ NetworkManager , NetworkManager HUD , Network Identity , Network Transform

public override void OnStartLocalPlayer() // 只针对一个客户端本地玩家做的回调

{

}

public override void OnStartServer() // 服务器启动时做处理

{

}

[Command] 命令 函数前也要加相对应的前缀 Cmd

该函数在服务器执行 在客户端被调用

eg. [Command]

Public void CmdFire(){}

spawn()函数 在连接到所有相同服务器的客户端共同创建一个游戏对象 状态实时更新 在服务器存放管理

eg. NetworkServer.Spawn(obj); // obj 为实例化的一个游戏对象

[ClientRpc] 命令 函数前也要加相对应的前缀 Rpc

该函数在服务器调用 在客户端执行

[SyncVar]

[SyncVar(hook = "ChangeHP")] // 通过 hook 指定对应函数 当服务器改变 playerHp 值, 客户端则会调用 ChangeHP() 函数

public int playerHp = 100;

public void ChangeHP(int \_hp)

{

playerHp = \_hp;

}

@ NavMesh

Off Mesh Link 组件

Navigation 中 Bake 相关参数： Agent Radius -- 代理半径 决定了地形烘焙出来后角色占地半径大小

Agent Height -- 代理高度 决定了具有头顶障碍区域烘焙出来后角色是否能通过

Max Slope -- 最大角度 决定了地形烘焙出来后角色可通过的地形坡度的最大角度

Drop Height -- 跳跃的最大高度 决定了地形烘焙出来后角色能否从相应位置跳下

Jump Distance -- 跳跃的最大距离 决定地形烘焙出来后角色能否进行跳跃通过断层区域

@ RequireComponent -- 当一个obj被挂上该脚本时 会自动挂上 require 的其他脚本

比如 在 A.cs 脚本中 添加 [RequireComponent(typeof(Animator))] 那么挂上 A 脚本同时 animator 组件也会被挂上

自动挂上去的脚本不能被移除...

[SerializeField] float m\_Speed; 强制序列化一个私有域 这样一个私有的变量 在 inspector视图面板上也能看得见

@ Unity 中 3 中 message 的消息管理

BroadcastMessage ------- 向挂了该脚本的obj及其所有子物体发送消息 调用指定函数

SendMessage ------- 向挂了该脚本的obj的所有脚本发送消息 调用指定函数

SendMessageUpwards ------- 向挂了该脚本的obj及其所有父物体发送消息 调用指定函数

\*\*\*\* Socket TCP 通信 \*\*\*\*

服务端：

1.创建套接字 Socket s = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

2.绑定IP和端口 s.Bind(new IPEndPoint(IPAddress.Parse(ip), port)); 这里的ip 和 port为传入的对应数据

3.监听客户端的连接 s.Listen(100); 参数对应能连接最大客户端数量

4.客户端连接后，则接受并返回对应套接字 Socket mClientSocket = s.Accept(); 该函数返回一个 Socket 类型的数据 之后的消息发送接受就通过 mClientSocket 处理

5.使用返回的套接字处理通信 Send() Receive()

可以开启两个线程 分别监听 客户端的连接请求 和 客户端消息的发送 Thread t = new Thread();

客户端：

1.创建套接字 保证IP地址和端口 同服务器一致 Socket c = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

2.连接服务器 c.Connect()

3.与服务器通信 Send()/Receive();

同样可以开启两个线程 分别监听 客户端的连接情况 和 消息发送

@ 关于锚点 和 中心点

UI 中心点 pivot (不是指 物体的中心 而是指 旋转的中心 )

取值范围 0 - 1 (0,0)在物体左下角 (0.5,0.5)物体中心 (1,1)物体右上角

一个控件的子节点锚点范围：

当对一个节点的子节点进行锚点设置的时候 子节点锚点的控制范围只能是在父节点的控件区域内