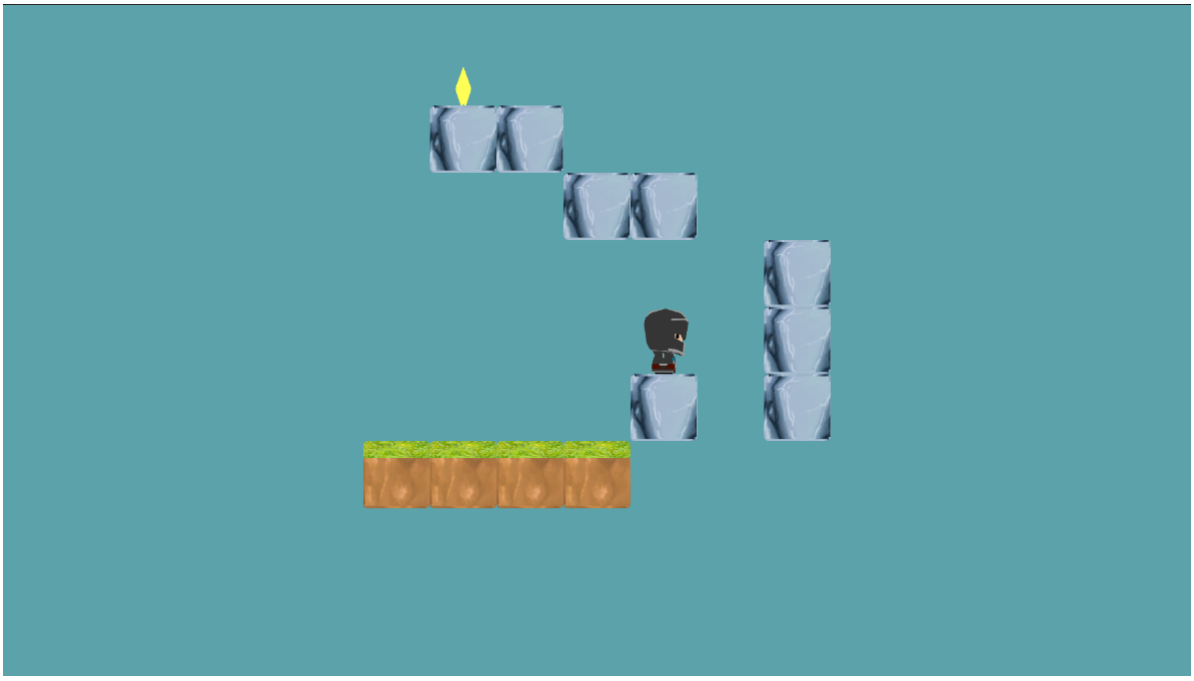
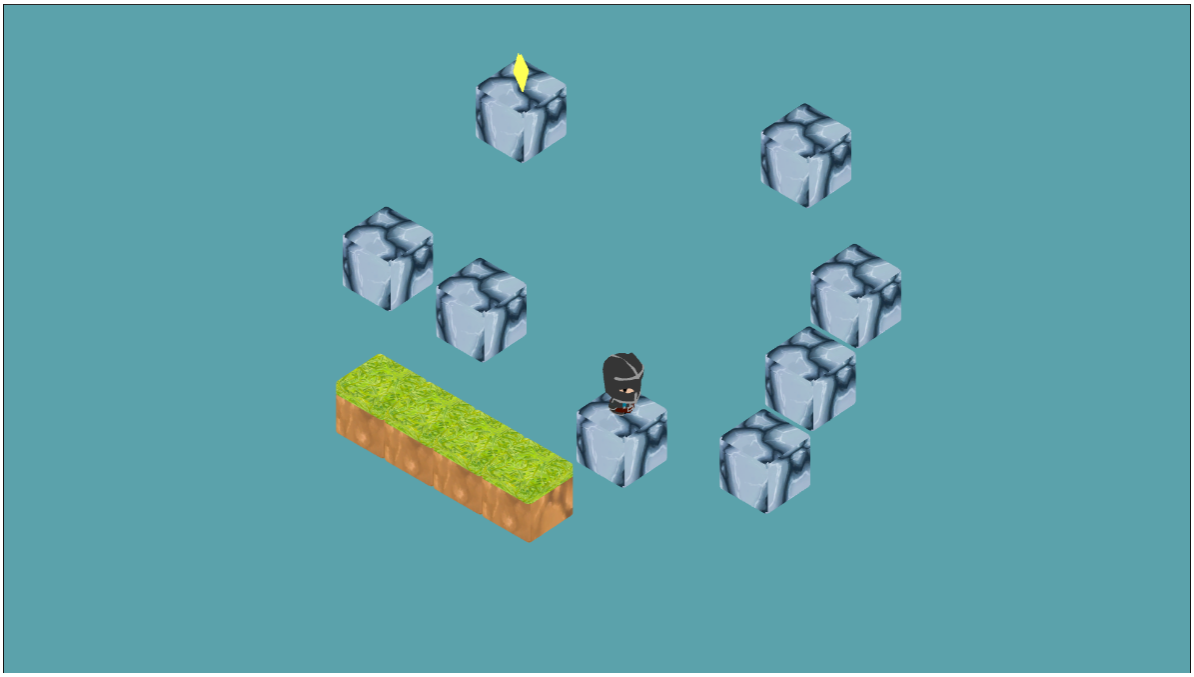
视角转动解谜游戏Demo

通过对看向场景的视角进行转动使得3D游戏世界转换成2D平面空间并由此进行解谜





摄像机转动

将3D世界呈现为2D平面首先要使得摄像机能够进行转动

并且需要将整个场景保持在摄像机的中央

可以通过获取组成场景方块的边界位置来得到整个场景的中点

```
public Vector3 GetCenterPoint()
{
    // 所有的方块放在 cubes 这个 gameobject 下
    Transform[] transforms = cubes.GetComponentsInChildren<Transform>();
    radius = GetCubeRadius(transforms[1]);

    for (int i = 1; i < transforms.Length; i++)
    {
        Transform trans = transforms[i];

        maxPos.x = Mathf.Max(maxPos.x, trans.position.x);
        maxPos.y = Mathf.Max(maxPos.y, trans.position.y);
        maxPos.z = Mathf.Max(maxPos.z, trans.position.z);
        minPos.x = Mathf.Min(minPos.x, trans.position.x);
        minPos.y = Mathf.Min(minPos.y, trans.position.y);
        minPos.z = Mathf.Min(minPos.z, trans.position.z);
    }

    return (maxPos + minPos) / 2;
}
```

由此可以得到整个场景的中心点，再通过设置一个合理的偏移值使得摄像机的位置不会和场景内物体重叠即可得到一个将整个场景置于中心的画面

```
public void RotateCam(RotationType currentType)
{
    Vector3 camRotation = new Vector3();
    Vector3 camPos = new Vector3();

    switch (currentType)
    {
        case RotationType.Front:
            camPos = new Vector3(centerPos.x, centerPos.y, minPos.z - viewOffset
* radius); // 设置一个合理的偏移值
            break;
        // ...
    }
```

当想要摄像机面对不同的角度时，可通过给定不同的偏移量再旋转摄像机的角度即可

这里可以使用线性插值来做出一个摄像机缓慢从当前位置到目标位置移动的效果

```
Vector3 lerpVal = Vector3.Lerp(Camera.main.transform.position, targetPos,
Time.deltaTime * camTransSpeed);
Camera.main.transform.position = lerpVal;

Vector3 targetAngle = new Vector3(
                Mathf.LerpAngle(Camera.main.transform.rotation.eulerAngles.x,
targetRotation.x, Time.deltaTime * camTransSpeed),
                Mathf.LerpAngle(Camera.main.transform.rotation.eulerAngles.y,
targetRotation.y, Time.deltaTime * camTransSpeed),
                Mathf.LerpAngle(Camera.main.transform.rotation.eulerAngles.z,
targetRotation.z, Time.deltaTime * camTransSpeed));

Camera.main.transform.eulerAngles = targetAngle;
```

设置碰撞体

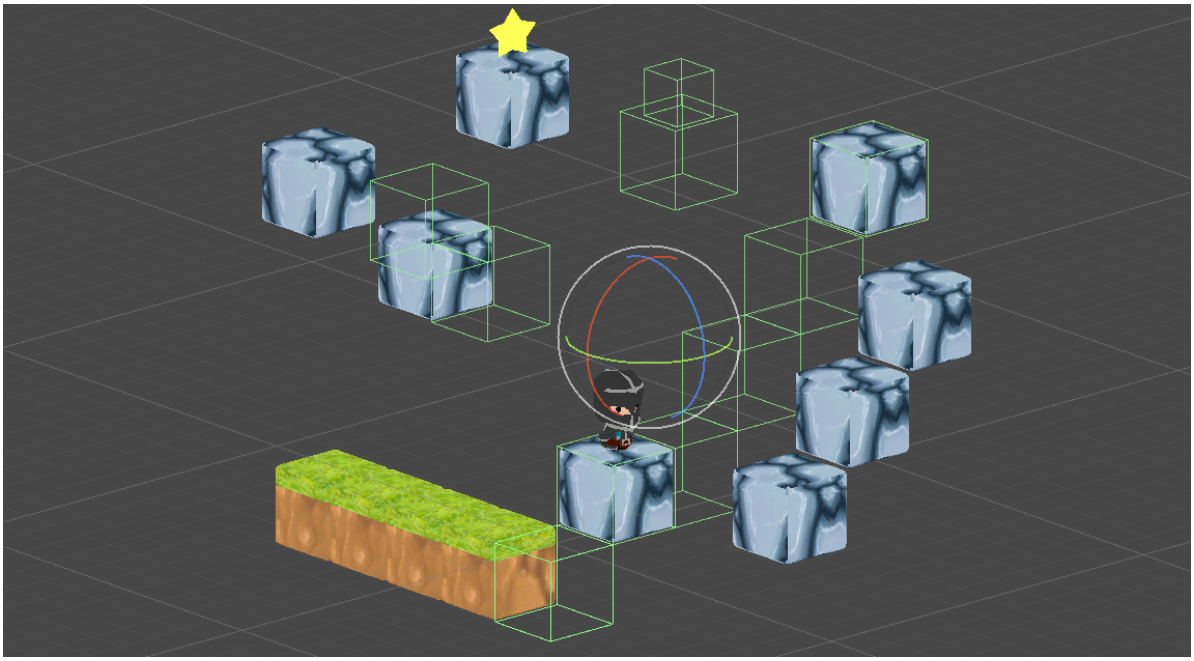
视角解谜的关键在于，无论3D空间中物体如何排列，在2D平面内玩家都是可以到达的

在通常情况下，摄像机转动后，场景内的物体的碰撞体的位置应该维持不变

但是如果想要实现视角解谜的效果，就需要在每次旋转后将所有碰撞体的位置更新到同一个平面内

这样，从视觉角度上来看，3D空间中不同位置的物体处于同一个平面

而和玩家进行交互的其实是每个物体的碰撞体



当游戏开始时，我们可以为场景内每个方块创建一个碰撞体，并将这些碰撞体和碰撞体的初始位置保存在`list`中

```
public void GenerateColliders()
{
    Transform[] transforms = cubes.GetComponentsInChildren<Transform>();
    colliderPos = new Vector3[transforms.Length];
    colliders = new Transform[transforms.Length];

    for (int i = 1; i < transforms.Length; i++)
    {
        // Create a new collider object and add into collider list
        GameObject newCollider = Instantiate(colliderPrimitive);
        newCollider.transform.position = transforms[i].position;
        newCollider.transform.parent = colliderList.transform;

        colliders[i - 1] = newCollider.transform;
        colliderPos[i - 1] = newCollider.transform.position;
        //newCollider.gameObject.layer = 10;
    }

    // ....
}
```

同时，当每次摄像机进行转动时，我们都需要更新碰撞体的位置来将碰撞体置于同一平面内

由于正面和反面是同一平面，同理右侧和左侧视角也为同一平面，所以我们只需要针对两种情况进行判断

```
public void RotateCollider(RotationType currentRotation)
{
    // Rotate colliders around Player current position
    Vector3 currentCollider = Player.instance.GetCurrentCollider();

    for (int i = 0; i < colliders.Length; i++)
    {
        if (currentRotation == RotationType.Front || currentRotation ==
RotationType.Back)
```

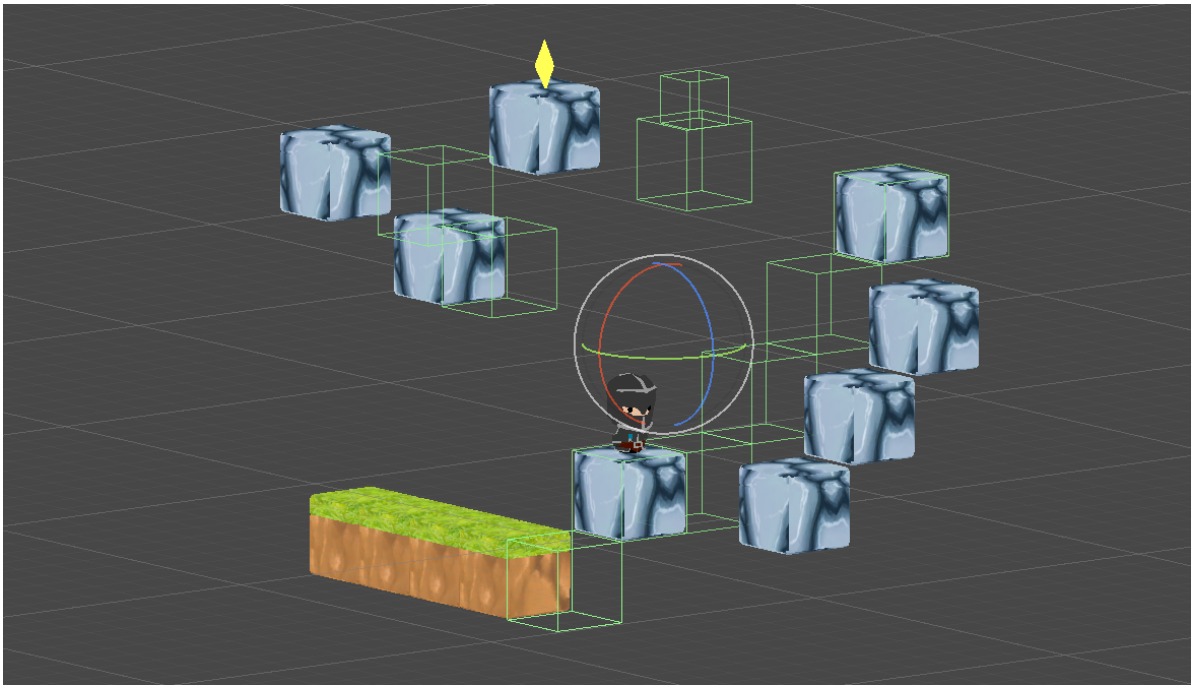
```
        {
            colliders[i].position = new Vector3(colliderPos[i].x,
colliderPos[i].y, currentCollider.z);
        }
        else if (currentRotation == RotationType.Left || currentRotation ==
RotationType.Right)
        {
            colliders[i].position = new Vector3(currentCollider.x,
colliderPos[i].y, colliderPos[i].z);
        }
    }

    //Update player to correct position
}
```

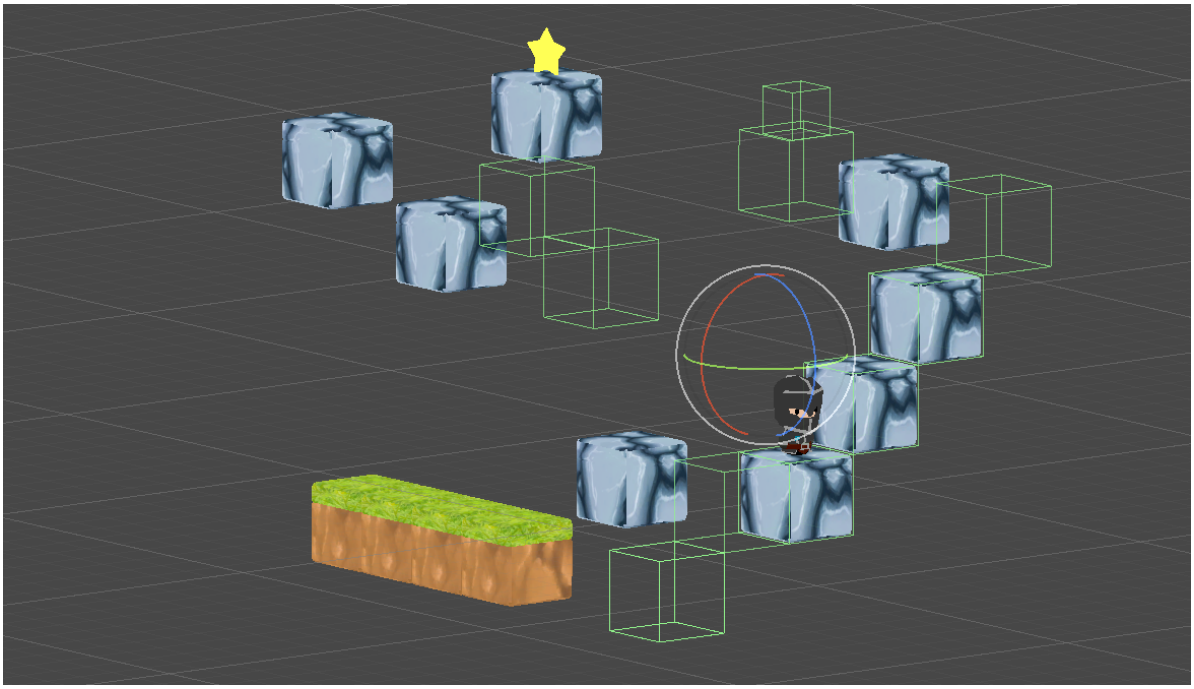
当摄像机接近于目标位置时，我们即可对碰撞体的位置进行更新

玩家移动

当玩家移动时，虽然视角内玩家是在平面内移动，但是场景内玩家在移动后需要被更新到方块的3D位置上

此时，我们需要根据玩家的碰撞体得到其原本方块所在的位置，同时对玩家位置进行更新





```
public void ResetColliderPos(Transform collider)
{
    for (int i = 0; i < colliders.Length; i++)
    {
        if (collider == colliders[i])
        {
            // Find the current collider
            // Reset its position
            if (collider.position != colliderPos[i])
            {
                collider.position = colliderPos[i];
            }
            break;
        }
    }
    RotateCollider(currentRotation);
}
```

思考

1. 由于在不同的视角下会出现方块重叠的情况，比如正向看是四个方块并排，而切换到左侧或右侧会出现只有一个方块显示的情况；这种情况下，所有方块的碰撞体将会重叠，而当玩家移动并返回这些方块时，由于重叠并不能获取玩家原本所在的方块，会出现视觉上的偏差
2. 在关卡设计上，由于视角转动的局限性，会出现转动时方块出现在同一平面并遮挡住玩家的情况，所以需要设定特定的位置进行转动