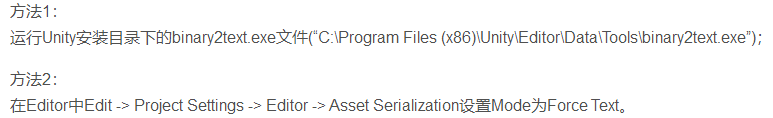
**Prefab**

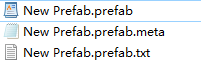
Prefab是一种可被重复使用的游戏对象

1. 它可以被置入多个场景中，也可以在一个场景中多次置入
2. 当你在一个场景中增加一个Prefabs，你就实例化了一个Prefabs
3. 所有Prefabs实例都是Prefab的克隆，所有如果实在运行生成对象会有（Clone）的标记
4. 只要Prefabs原型发生变化，所有的Prefabs实例都会产生变化

如何打开prefab文件：

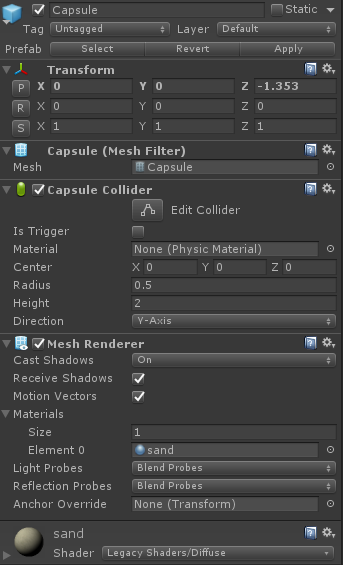


采用binary2text.exe方式，仅仅只需要将文件拖到该执行文件即可：



预设，类似各种UI编辑器编辑后的输出文件，unity中几乎任何事物都可以打包成预设，然后通过外部文件的形式再加载进程序中，不过是png/jpg等图片图集资源；GameObject Chartater之类的对象资源

一般的预设并不是直接把png、jpg合并成一个文件，而是添加他们的引用，预设与其说是UI界面等的带出文件生成文件，不如直接说是UI界面等的描述文件，他描述了某个界面引用了哪个图片，按钮默认状态是什么，按钮显示Title是什么，比如下面的就是看到的prefab文件：



%YAML 1.1

%TAG !u! tag:unity3d.com,2011:

--- !u!1001 &100100000

**Prefab:**

m\_ObjectHideFlags: 1

serializedVersion: 2

m\_Modification:

m\_TransformParent: {fileID: 0}

m\_Modifications: []

m\_RemovedComponents: []

m\_ParentPrefab: {fileID: 0}

m\_RootGameObject: {fileID: 1000011620259008}

m\_IsPrefabParent: 1

--- !u!1 &1000011620259008

**GameObject:**

m\_ObjectHideFlags: 0

m\_PrefabParentObject: {fileID: 0}

m\_PrefabInternal: {fileID: 100100000}

serializedVersion: 4

m\_Component:

- 4: {fileID: 4000012997215244}

- 33: {fileID: 33000010296479690}

- 136: {fileID: 136000011958227798}

- 23: {fileID: 23000013745129830}

m\_Layer: 0

m\_Name: Capsule

m\_TagString: Untagged

m\_Icon: {fileID: 0}

m\_NavMeshLayer: 0

m\_StaticEditorFlags: 0

m\_IsActive: 1

--- !u!4 &4000012997215244

**Transform:**

m\_ObjectHideFlags: 1

m\_PrefabParentObject: {fileID: 0}

m\_PrefabInternal: {fileID: 100100000}

m\_GameObject: {fileID: 1000011620259008}

m\_LocalRotation: {x: 0, y: 0, z: 0, w: 1}

m\_LocalPosition: {x: 0, y: 0, z: -1.353}

m\_LocalScale: {x: 1, y: 1, z: 1}

m\_LocalEulerAnglesHint: {x: 0, y: 0, z: 0}

m\_Children: []

m\_Father: {fileID: 0}

m\_RootOrder: 0

--- !u!23 &23000013745129830

**MeshRenderer:**

m\_ObjectHideFlags: 1

m\_PrefabParentObject: {fileID: 0}

m\_PrefabInternal: {fileID: 100100000}

m\_GameObject: {fileID: 1000011620259008}

m\_Enabled: 1

m\_CastShadows: 1

m\_ReceiveShadows: 1

m\_MotionVectors: 1

m\_LightProbeUsage: 1

m\_ReflectionProbeUsage: 1

m\_Materials:

- {fileID: 2100000, guid: a40a40a3e330dd64b88859d19e82b8ec, type: 2}

m\_SubsetIndices:

m\_StaticBatchRoot: {fileID: 0}

m\_ProbeAnchor: {fileID: 0}

m\_LightProbeVolumeOverride: {fileID: 0}

m\_ScaleInLightmap: 1

m\_PreserveUVs: 1

m\_IgnoreNormalsForChartDetection: 0

m\_ImportantGI: 0

m\_SelectedWireframeHidden: 0

m\_MinimumChartSize: 4

m\_AutoUVMaxDistance: 0.5

m\_AutoUVMaxAngle: 89

m\_LightmapParameters: {fileID: 0}

m\_SortingLayerID: 0

m\_SortingOrder: 0

--- !u!33 &33000010296479690

**MeshFilter:**

m\_ObjectHideFlags: 1

m\_PrefabParentObject: {fileID: 0}

m\_PrefabInternal: {fileID: 100100000}

m\_GameObject: {fileID: 1000011620259008}

m\_Mesh: {fileID: 10208, guid: 0000000000000000e000000000000000, type: 0}

--- !u!136 &136000011958227798

**CapsuleCollider:**

m\_ObjectHideFlags: 1

m\_PrefabParentObject: {fileID: 0}

m\_PrefabInternal: {fileID: 100100000}

m\_GameObject: {fileID: 1000011620259008}

m\_Material: {fileID: 0}

m\_IsTrigger: 0

m\_Enabled: 1

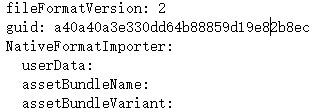
m\_Radius: 0.5

m\_Height: 2

m\_Direction: 1

m\_Center: {x: 0, y: 0, z: 0}

prefab中详细记录了每一项的信息，unity为每个资源都分配了一个guid，然后根据guid来引用各个资源，文件的guid存储于.meta文件中，比如上面的材质meta文件为：



我们发现上面的prefab中的红色部分正好是当前材质sand的guid，而且meta文件记录了assetbundle信息