# Generate rope via code

To create rope, you need:

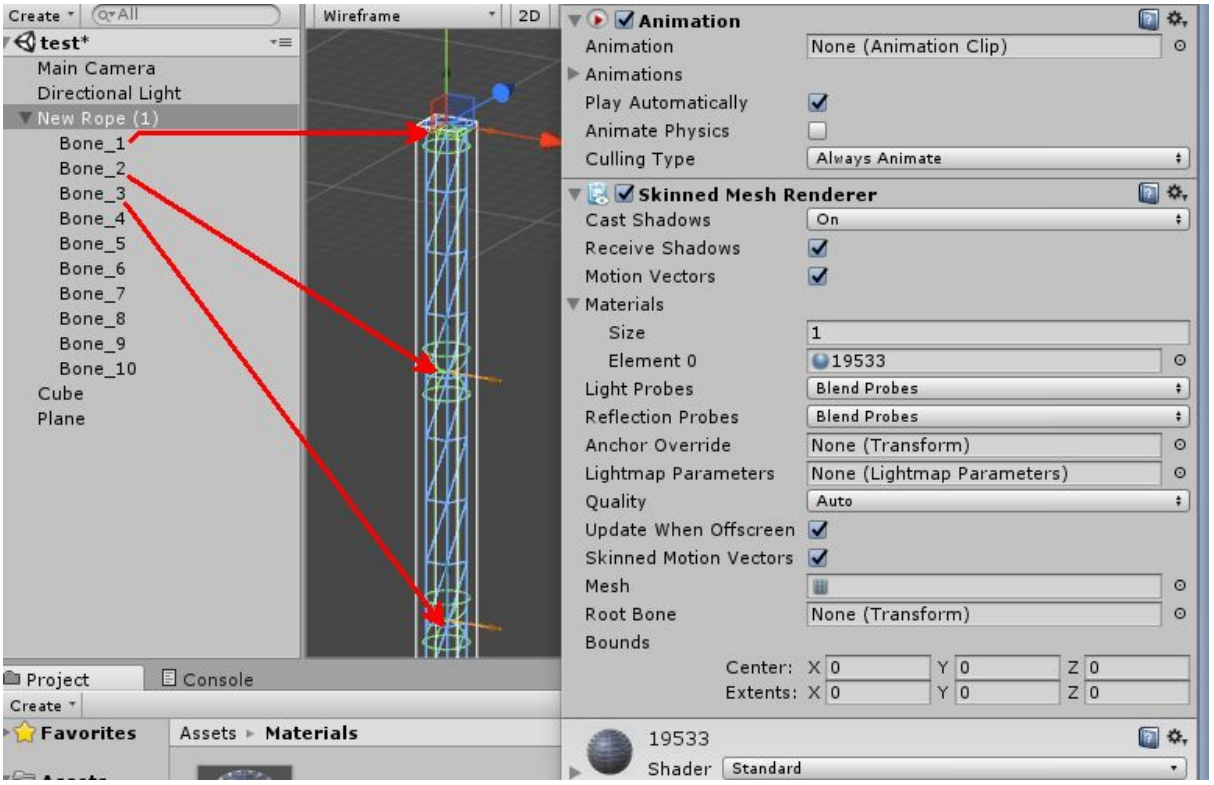
1) Create instance of mesh generator, see IRopeMeshGenerator
2) Create instance of RopeGenerator
3) Invoke method “MakeOne()”.

## RopeGenerator

Generator have only one constructor:
RopeGenerator(IRopeMeshGenerator meshGenerator, Material material, GameObject gameObject)
Where,

- gameObject is an object where bones will be added.

Result of generated rope:




You can specify some parameters before you invoke method “MakeOne()”:

- JointAngleLimit - max angle between two adjacent bone
- BoneCount - count of bones in a rope
- MassOfBone - mass of each bone
- Length - total length of rope
- Radius - radius of rope
- RestrictFirstBone - if false, JointAngleLimit and skeletal weights will not be applied to the first bone.

Method ‘MakeOne’ returns a sequence of bones.

# IRopeMeshGenerator

Interface with only one method:

```
/// <summary>
/// Mash generator for rope generator
/// </summary>
public interface IRopeMeshGenerator
{
        /// <summary>
        /// Create mesh
        /// </summary>
        /// <param name="radius">Radius of rope</param>
        /// <param name="length">Length of rope</param>
        /// <param name="boneCount">Bones the mesh consist of</param>
        /// <param name="restrictFirstBone">If false, JointAngleLimit and skeletal weights will not be applied to first bone</param>
        Mesh Create(float radius, float length, int boneCount, bool restrictFirstBone);
}
```

This method must create mesh and set weights.

## RopeMeshCylinderGenerator

RopeMeshCylinderGenerator - is implementation of IRopeMeshGenerator that generates a simple cylinder.

This class has only one parameterless constructor.
Some properties of RopeMeshCylinderGenerator that you can tweak:

- Sides - Sets the number of sides around the cylinder.
- SegmentsPerBone - Sets the number of divisions along the bone's major axis.

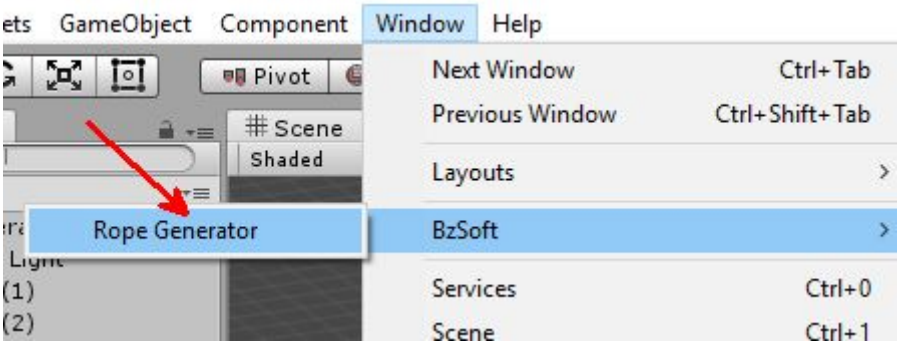
## RopeMeshChainGenerator

Generates a chain.
In example from the sample scene I used this parameters:

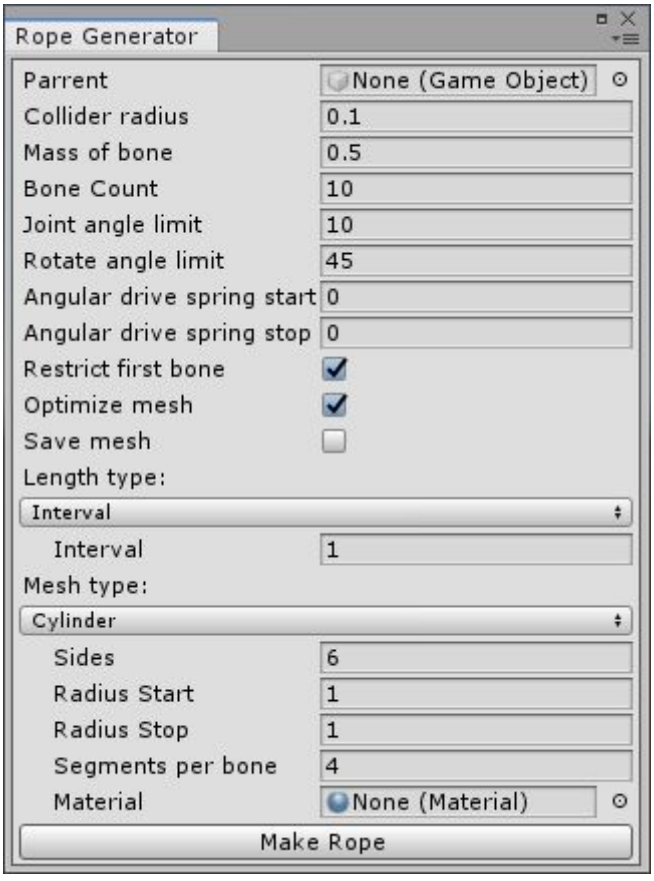
| | |
|---|---|
| collider radius | 0.45 |
| mass | 0.05 |
| joint angle limit | 90 |
| spring | 100 |
| interval | 1 |
| chains per bone | 2 |
| scale | 0.45 |
| initial rotation | 0, 0, 90 |
| step rotation | 90, 0, 0 |

# Generate rope via tool.

Open tool:




Dialog:




Here you can tweak any parameter and then click “Make rope”.
Description of properties see in “Generate rope via code” section.

# Create a prefab

To create a prefab you must save a mesh. Check “Save mesh” check box and press “Make Rope”. Then you can create a prefab of the rope in a normal way.

# Rope Physic helper

Sometimes you can see that the rope is not moving correctly if you move it very fast. To help it move correctly, attach the script RopePhysicHelperController.
Properties:

- Precision: maximum length mismatch.

# Replace mesh generator

If you need to use your one mesh generator, you need to create a new class that implements `IRopeMeshGenerator` interface and pass it to the constructor of `RopeGenerator`.

# Cylinder Example

```
// create main game object of your rope
var go = new GameObject("MyTestRope");

// prepare material for rope mesh
var material = new Material(Shader.Find("Diffuse"));

// create mesh generator
var meshGenerator = new RopeMeshCylinderGenerator();
// In line below, I set mesh to have 8 sides:
meshGenerator.Sides = 8;

// set start/stop mesh radius
meshGenerator.RadiusStart = 0.1f;
meshGenerator.RadiusStop = 0.1f;

// create rope generator
var rope = new RopeGenerator(meshGenerator, go);
rope.Length = 12;
rope.ColliderRadius = 0.1f;

// generate rope
GameObject[] bones = rope.MakeOne();
// you can use `bones` variable to get all generated bones.
// e.g., to attach sinker to the end of your rope (to last bone)
```