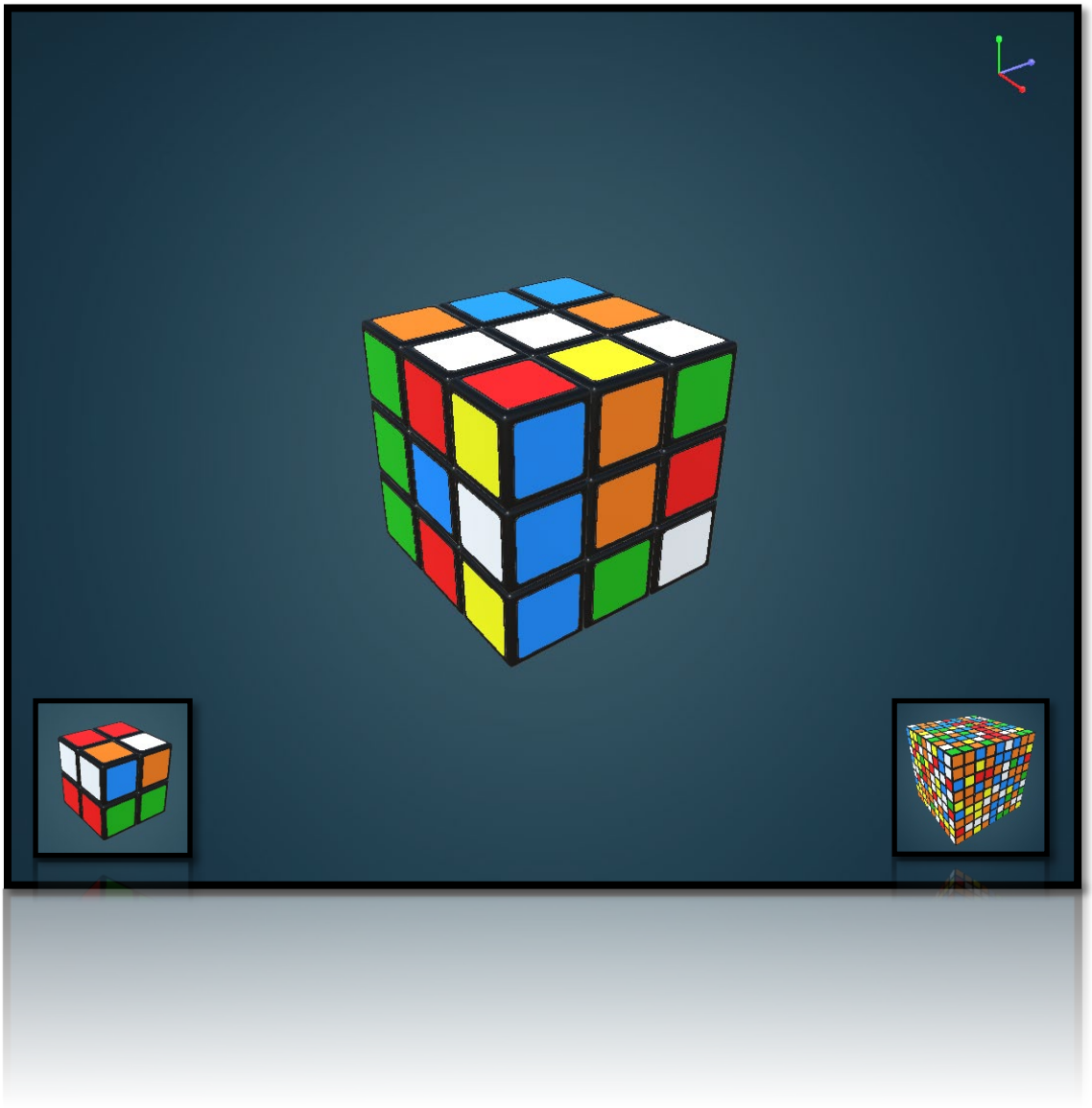
Interactive Rubik Cube (x * x * x)

Unity C# implementation by Inner Drive Studios



Introduction

Hey hello, welcome and thank you for buying this package, consisting of all the assets and code required to create an interactive Rubik Cube. This document describes the most important details of this package to help you get started quickly.

Package goals

This package has two main goals:

- Providing you with a starting point to build your own interactive Rubik cube application
- Act as a learning project for those interested in the 3D math behind implementing such a cube (note that reading the comments and understanding the sourcecode does require a fundamental understanding of vectors, matrices, dots, crossproducts and left & right handed coordinate systems).

The package provides an optimized implementation that allows for cubes of arbitrary sizes *without* the use of extensive ray casting (the provided sample scene demonstrates cube sizes 2 to 9, but you can go higher if you wish). In addition the package is divided in clear classes that can be reused as is, or adapted to your needs as you see fit.

Contents

- Clear, cohesive and extensively documented classes
- A UV-ed cublet mesh with realistic albedo, normal and smoothness textures (.psd included)
- A cublet.blend file with high and low poly cublet meshes included
- A sample scene demonstrating all the features

Most important classes and features

- RubikCube class: contains core cube definition, rotation and undo features and the ability to check if the cube is solved or not
- DiscRotator: transforms your mouse actions into disc rotation actions on the RubikCube
- CubeRotator: transforms your mouse actions into 90° rotations on the cube as a whole
- CameraMouseOrbit: transforms your mouse actions into free orbit camera movements
- AxisDisplay: shows the orientation of your cube on the top right of your screen as a gizmo
- RubikCubeInitializer: shows you how to spawn a cube of a specific size and handle its lifetime

Package limitations:

- This package doesn't come with a cube solving algorithm
- All the rubik cubes are built up dynamically using the same prefab for every cublet, in other words there is no distinction between corner, edge and center pieces.
- There is no side introspection mechanism included, in other words, the cube *can* tell you whether it is solved or not, but you can't say "Get me the color of cublet R(2,2)" for example.

Getting started

To get started quickly, please first open the provided RubikCubeScene, press play and try out all the things shown in the onscreen help. After that, click on every element in the scene and check out its 'Note' component with additional info.

The basic idea is that we have a RubikCubeInitializer which allows you to (re)spawn a RubikCube prefab instance. The spawned RubikCube instance can then be initialized with a specific size and a prefab from which all its cublets (the little cubies ;)) can be created. All the cube behaviour (managing the cublets, rotating discs and rotating the cube itself in 90 degree increments) is implemented through different scripts on the RubikCube instance.

Different cublet types & materials

When creating the Rubik cube you can specify which cublets to use.

There are two different cublets:

- a cublet with rounded edges and separate materials for each side and the cublet itself.
- a cublet with rounded edges and a single material for each side and the cublet itself.

Both amount to the same number of draw calls after batching and both use the same underlying (and UV'ed) mesh. If you don't want to customize it, it doesn't matter which one you use and if you do want to customize you can pick and choose whether you like the separate material route better or the single material one (for example in case you'd like to use substance painter, in which case you should also check the high poly cube version in the provided blendfile).

The layout of the cublet is important, since the cube provides an option to turn any invisible sides off, so *if* you want to customize the cublet meshes in any way, make sure you check the layout on one of the provided example cublets.

Questions?

Drop me a line at info@innerdrivestudios.com