Solving The Rubik's Cube

A Reflective Practice

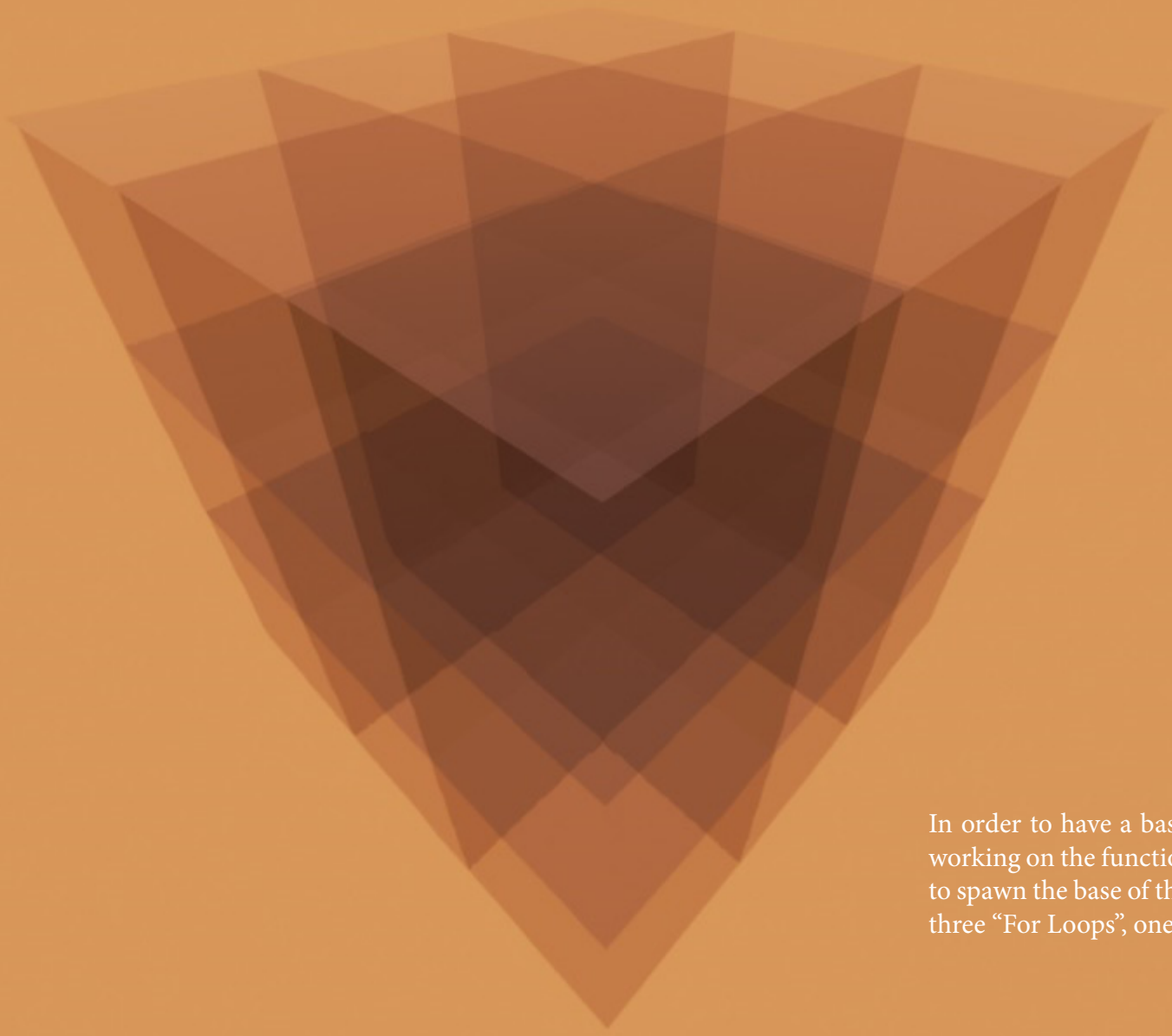
by Hien Quy Tran

Date: 11/11/2014
Student Number: 543800

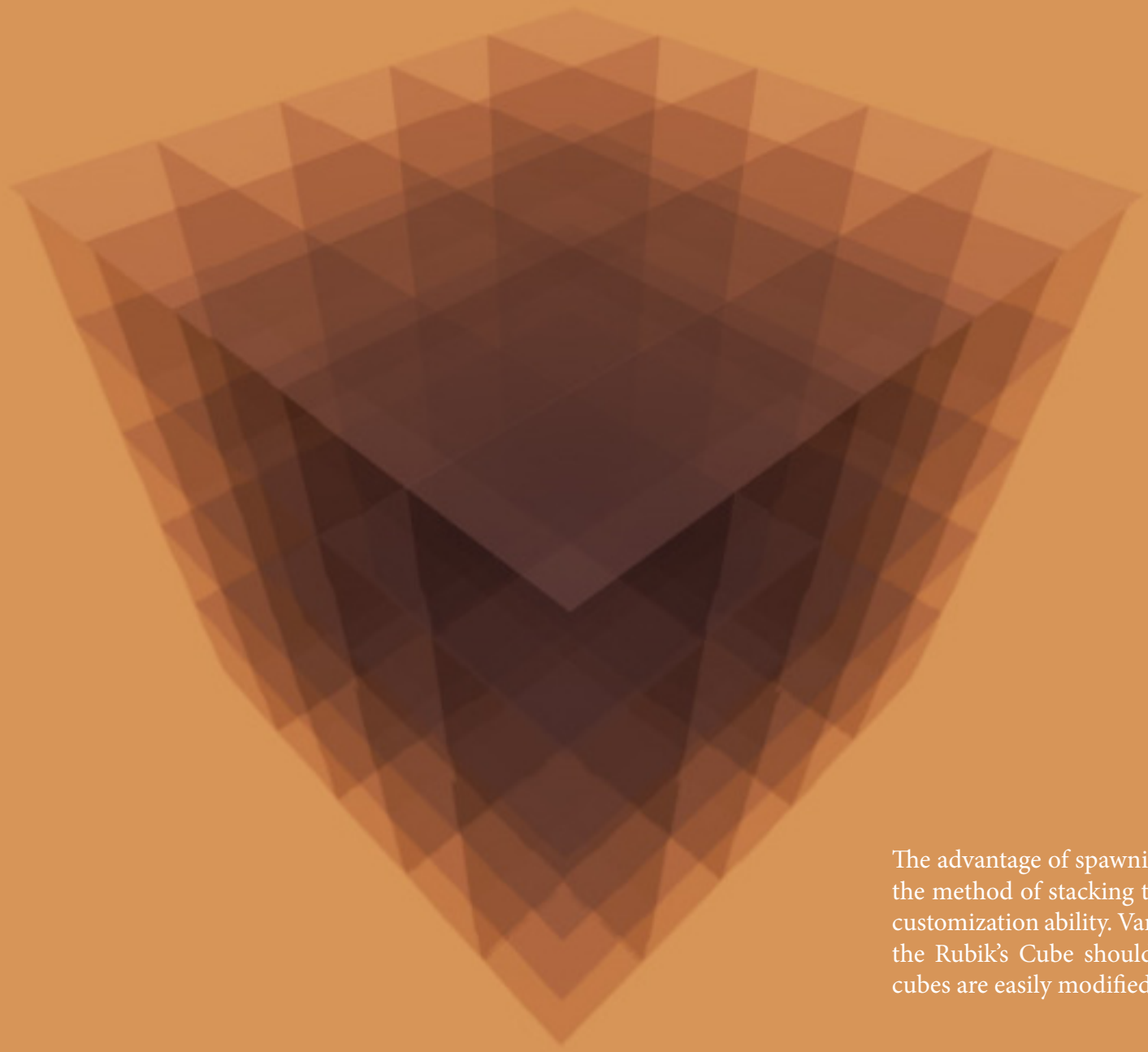
Lecture: Game Design
3rd Semester
Project: Development of a Modular Game in Unreal Engine 4

Constructing the Base

Approximate Working Hours: 1



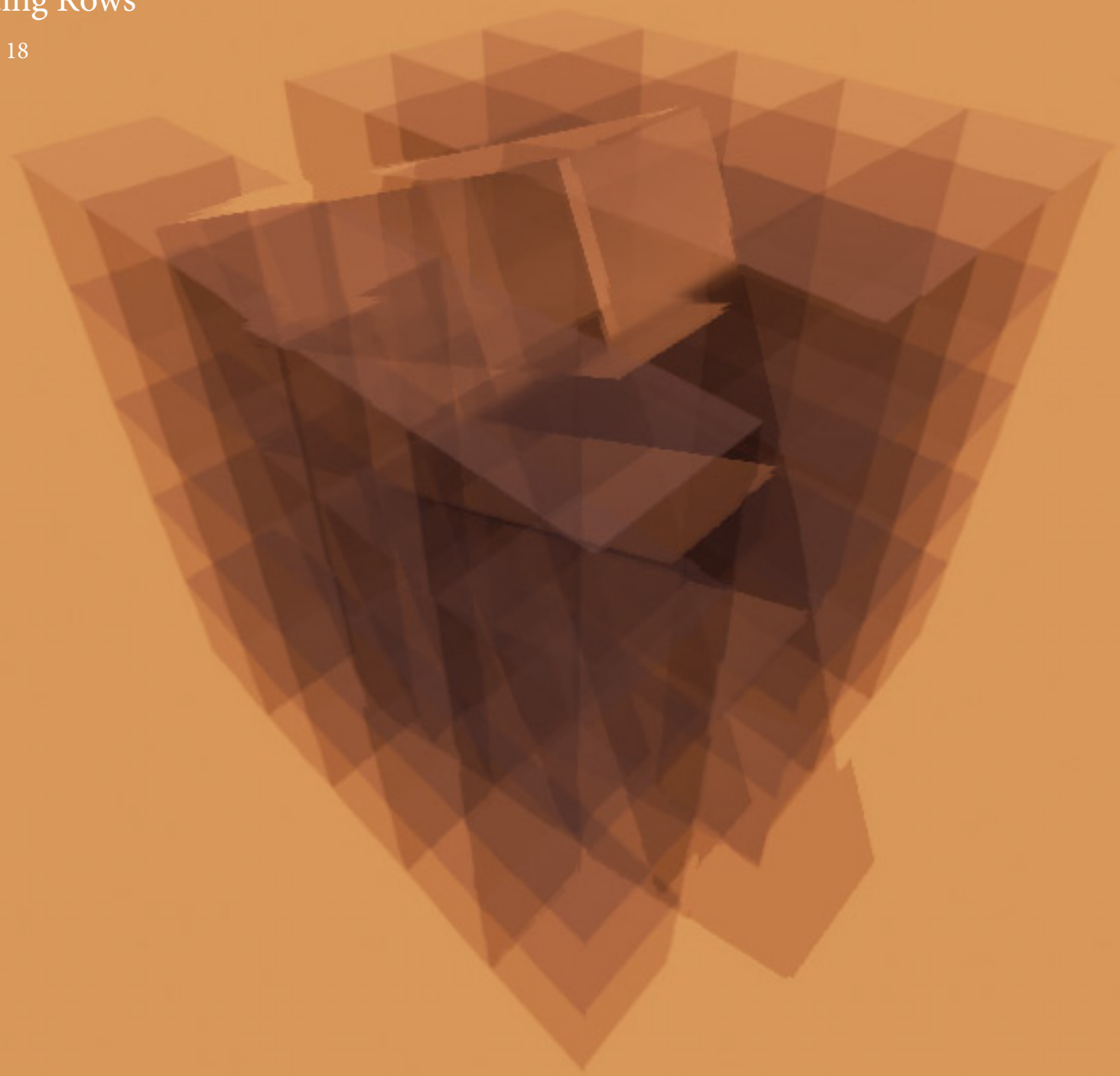
In order to have a basic structure from where I could start working on the functionalities of the Rubik's Cube, I decided to spawn the base of the Rubik's Cube itself via code by using three "For Loops", one for each dimension.

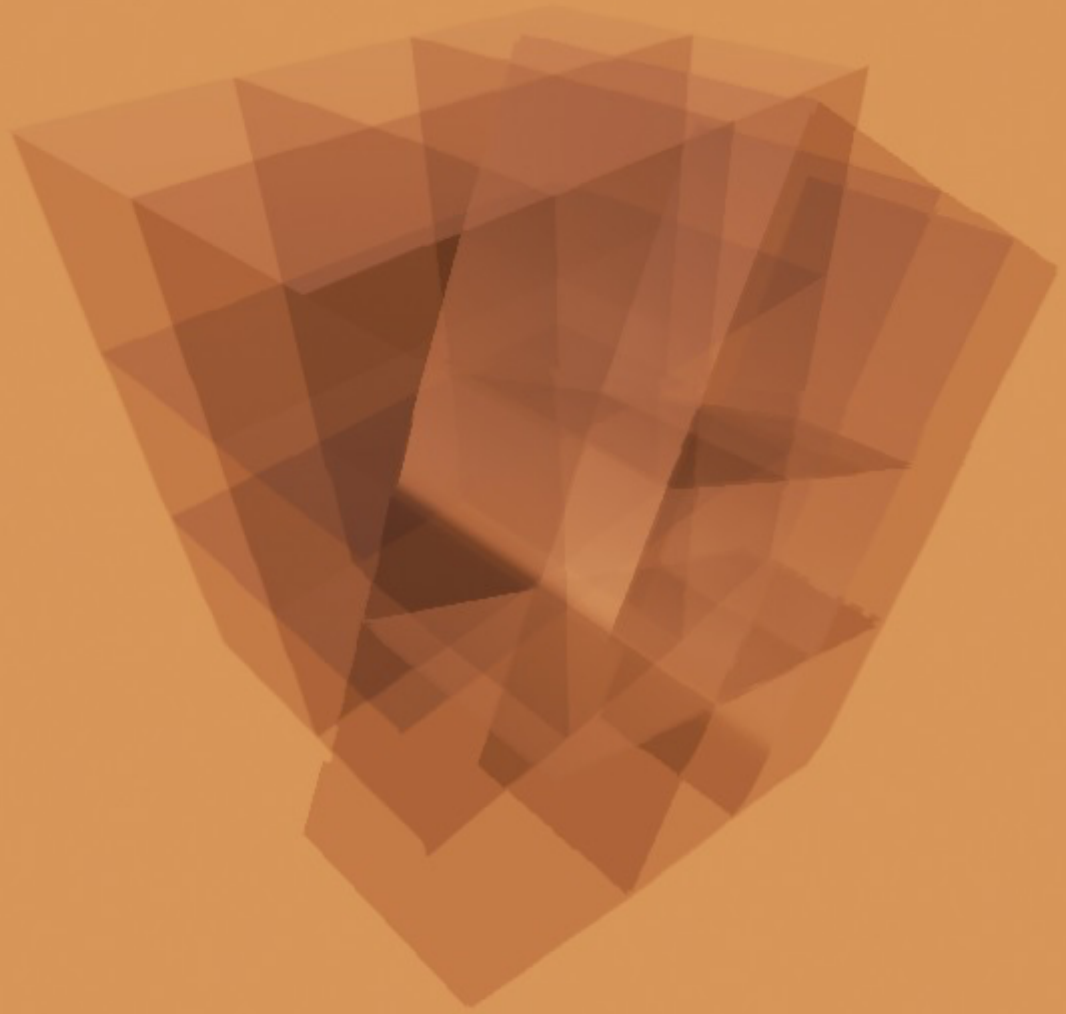


The advantage of spawning the Rubik's Cube via code, compared to the method of stacking the cubes together by hand, lies in the easy customization ability. Variations like the number of how many cubes the Rubik's Cube should consist of or the scale of the individual cubes are easily modified by changing some variables.

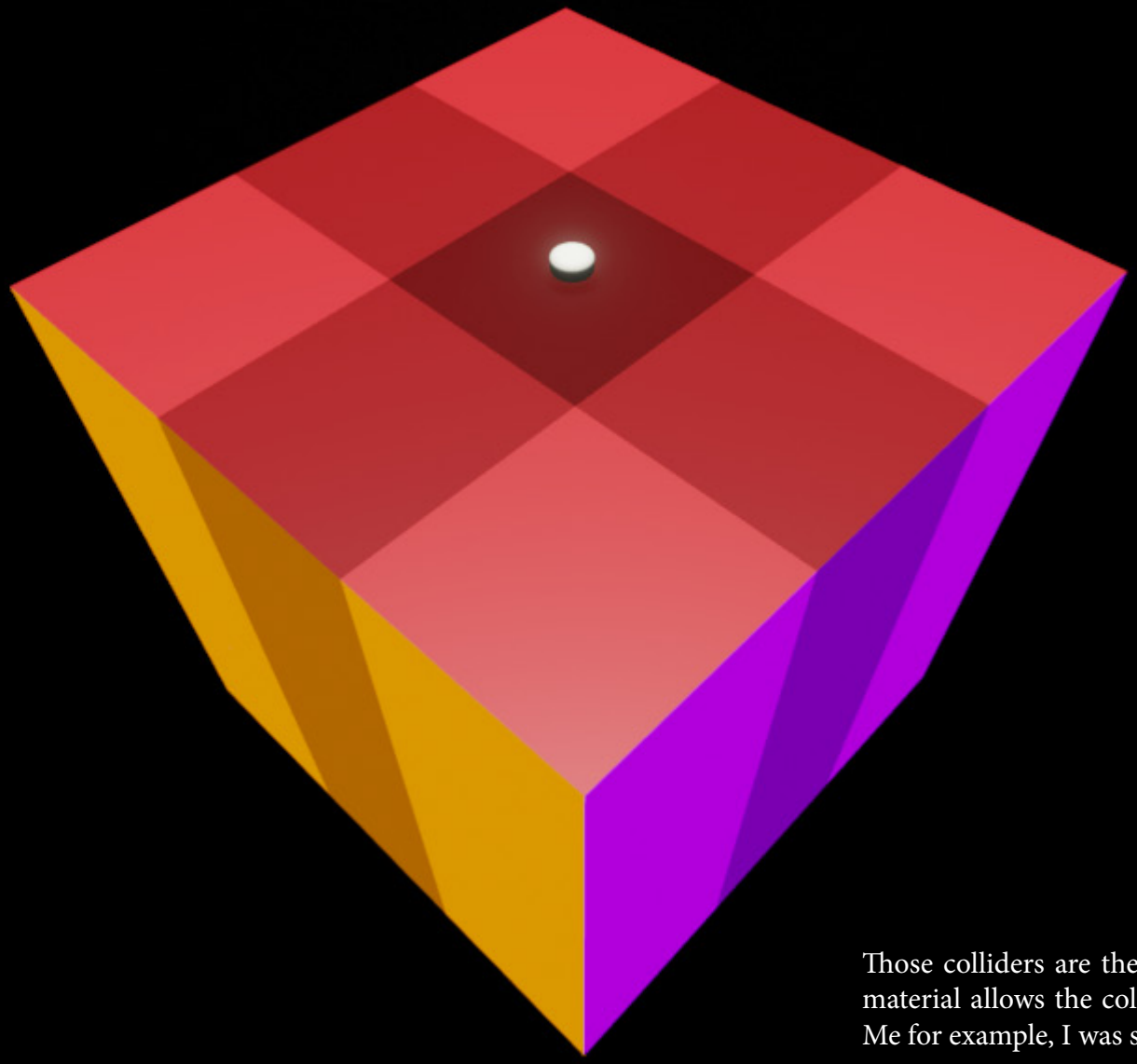
Grouping and Rotating Rows

Approximate Working Hours: 18

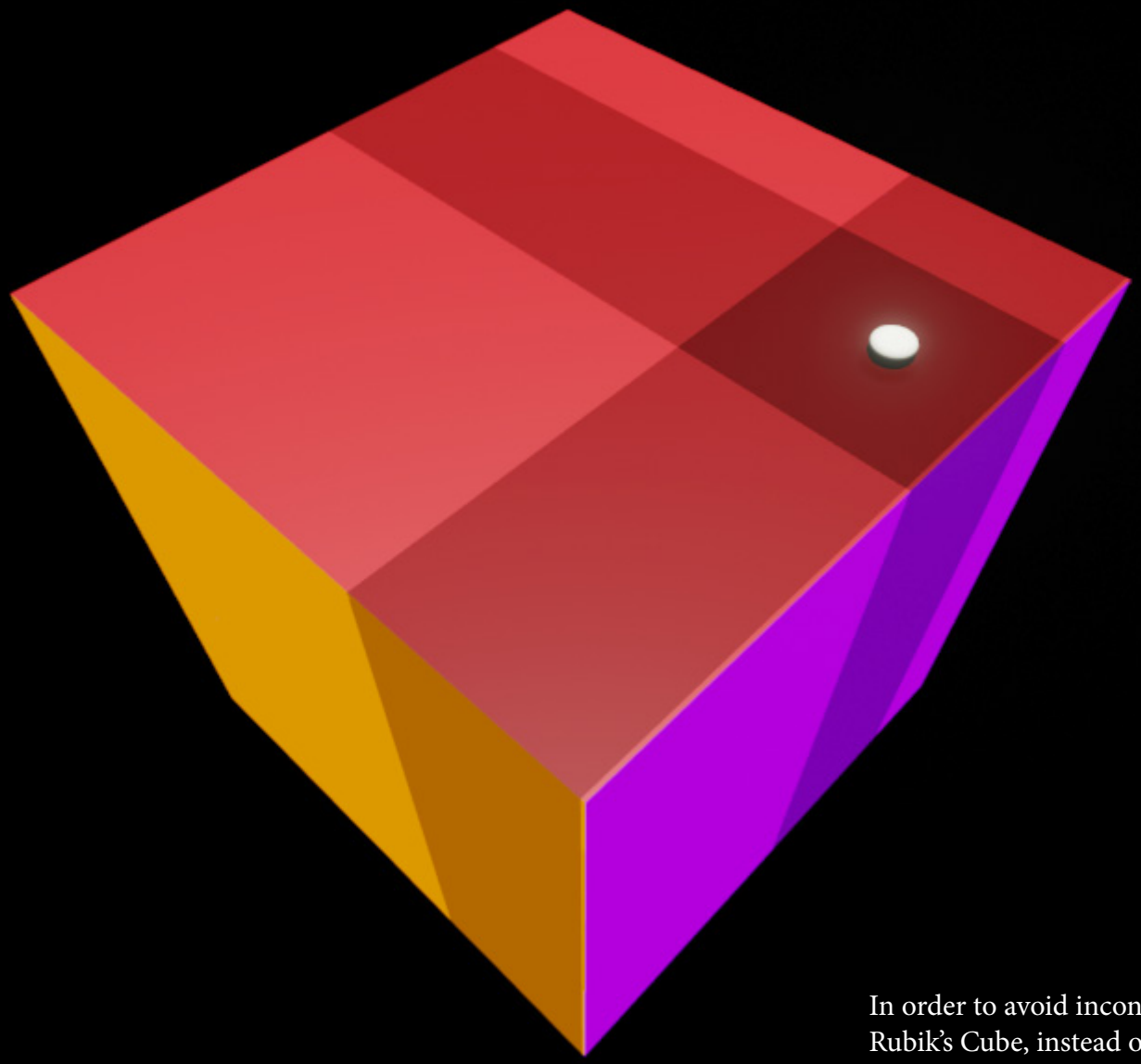




Having the basic structure of the Rubik's Cube with its individual cubes, the next task was to find a solution of how to group a row of cubes together in order to rotate them. For that, I spawned two colliders, which are adapting to the width of the Rubik's Cube.

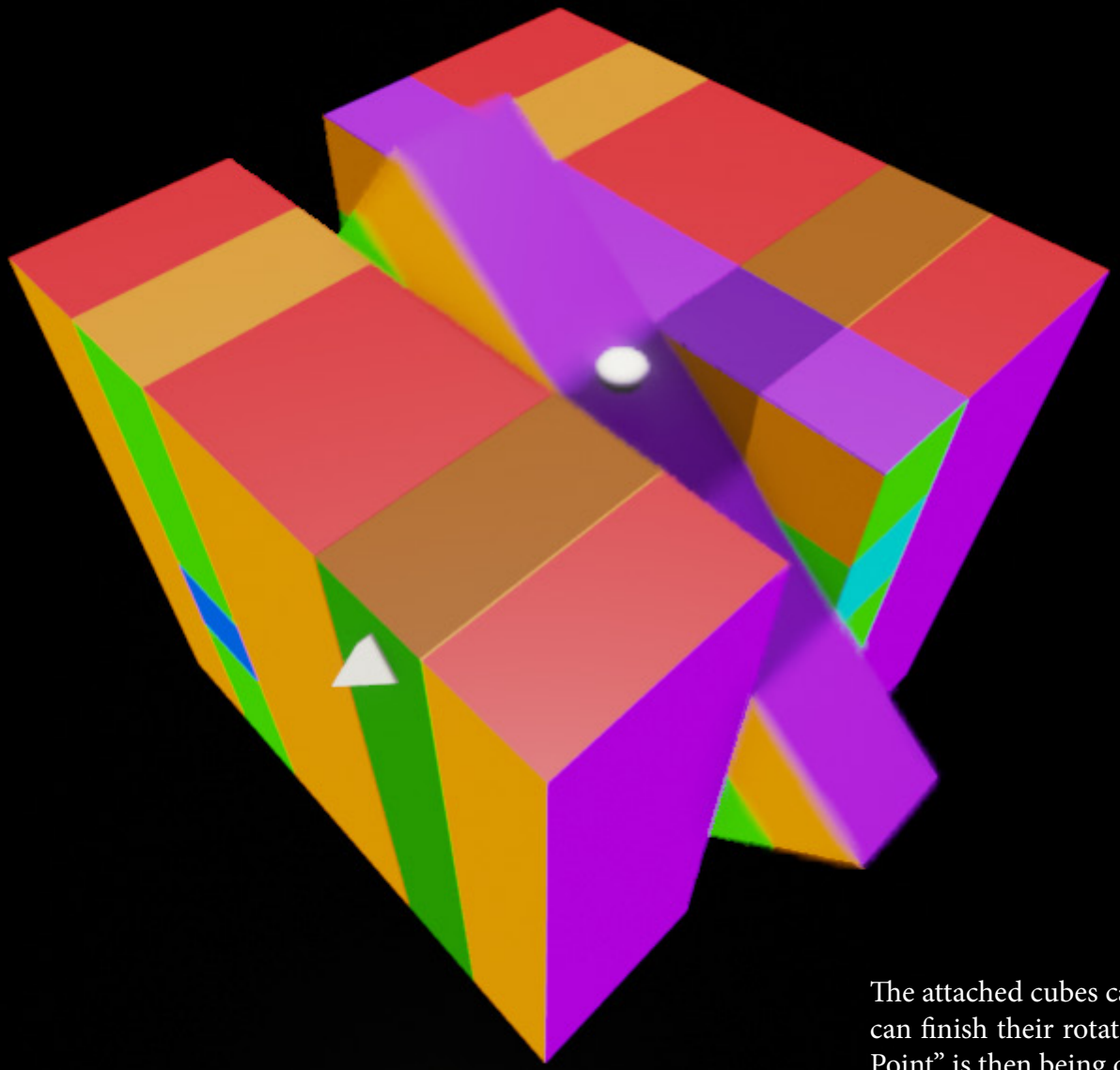


Those colliders are then being linked with a custom material. The material allows the colliders to be customized in their appearance. Me for example, I was setting the material to gray translucent.



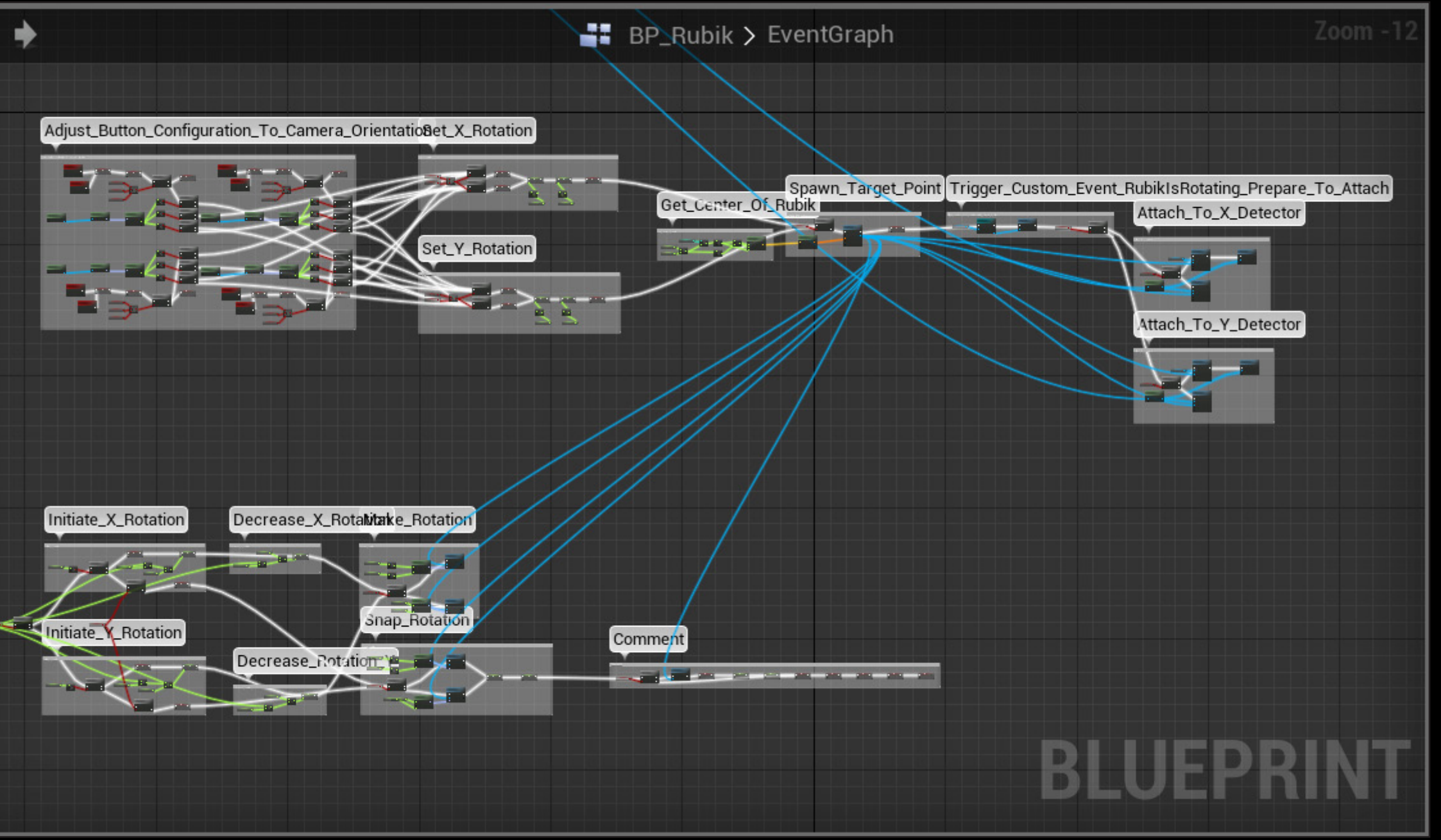
In order to make those colliders movable, I decided to create a cursor. The cursor is being spawned on top of the Rubik's Cube and the area, in which it is allowed to move, is adapting to the size of the Rubik's Cube. Each collider for the rotation is then following the cursor along one axis.

In order to avoid inconsistencies, I needed the colliders to snap to the grid of the Rubik's Cube, instead of following the cursor seamlessly between the cube rows.



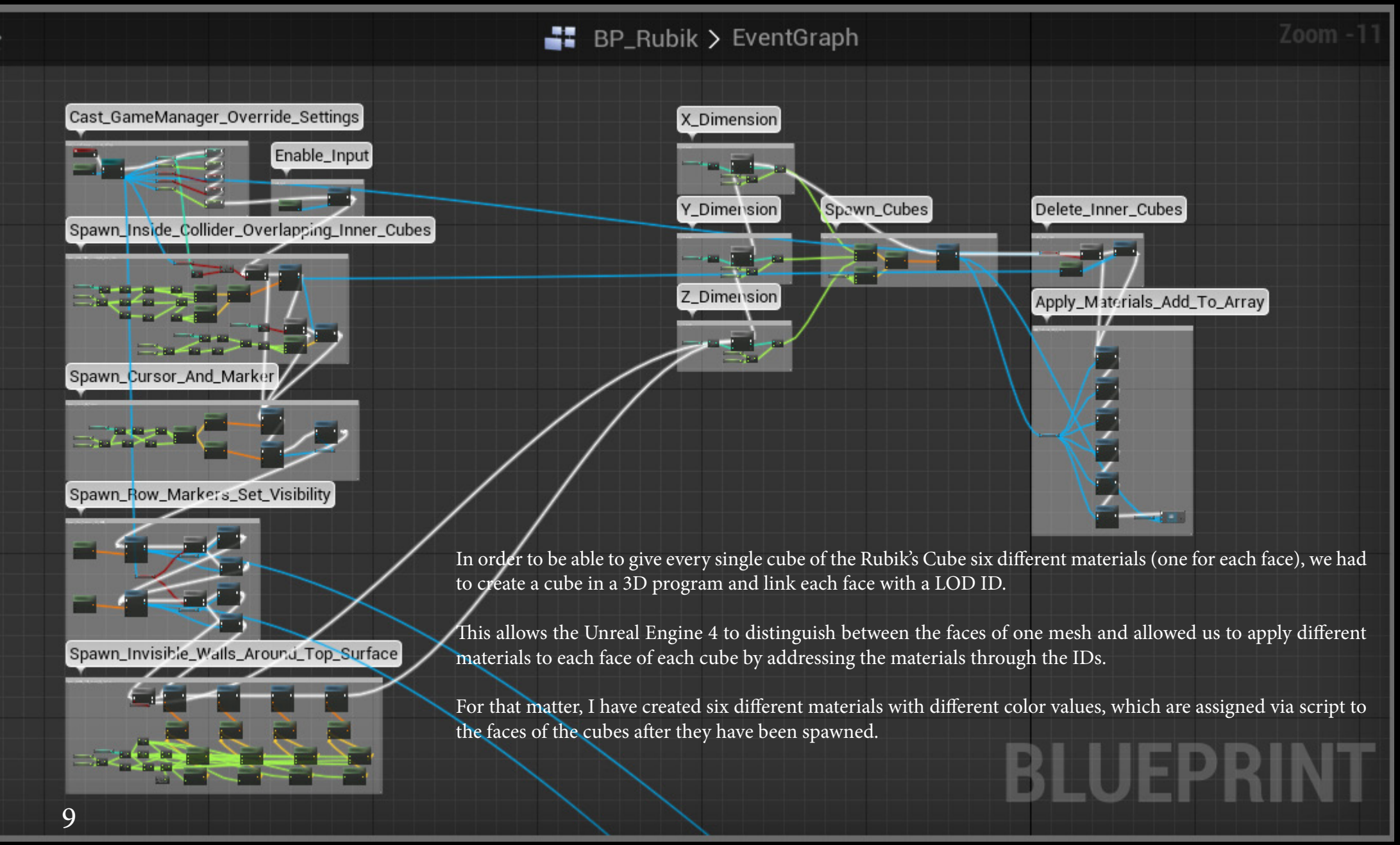
Having the grid snapping done, I was then able to group all cubes together, which are overlapping the collider, by attaching them to a “Target Point”, which is instantly spawned in the center of the Rubik’s Cube before every rotation.

The attached cubes can then be rotated around their “Target Point” and can finish their rotation with a snapping to a 90° rotation. The “Target Point” is then being destroyed in order to detach the cubes again.


BP_Rubik > EventGraph
Zoom -12
Adjust_Button_Configuration_To_Camera_Orientation
Set_X_Rotation
Set_Y_Rotation
Get_Center_Of_Rubik
Spawn_Target_Point
Trigger_Custom_Event_RubikIsRotating_Prepare_To_Attach
Attach_To_X_Detector
Attach_To_Y_Detector
Initiate_X_Rotation
Decrease_X_Rotation
Make_Rotation
Initiate_Y_Rotation
Snap_Rotation
Decrease_Rotation
Comment
BLUEPRINT

Applying Materials

Approximate Working Hours: 2



In order to be able to give every single cube of the Rubik's Cube six different materials (one for each face), we had to create a cube in a 3D program and link each face with a LOD ID.

This allows the Unreal Engine 4 to distinguish between the faces of one mesh and allowed us to apply different materials to each face of each cube by addressing the materials through the IDs.

For that matter, I have created six different materials with different color values, which are assigned via script to the faces of the cubes after they have been spawned.

Creating a Game Manager

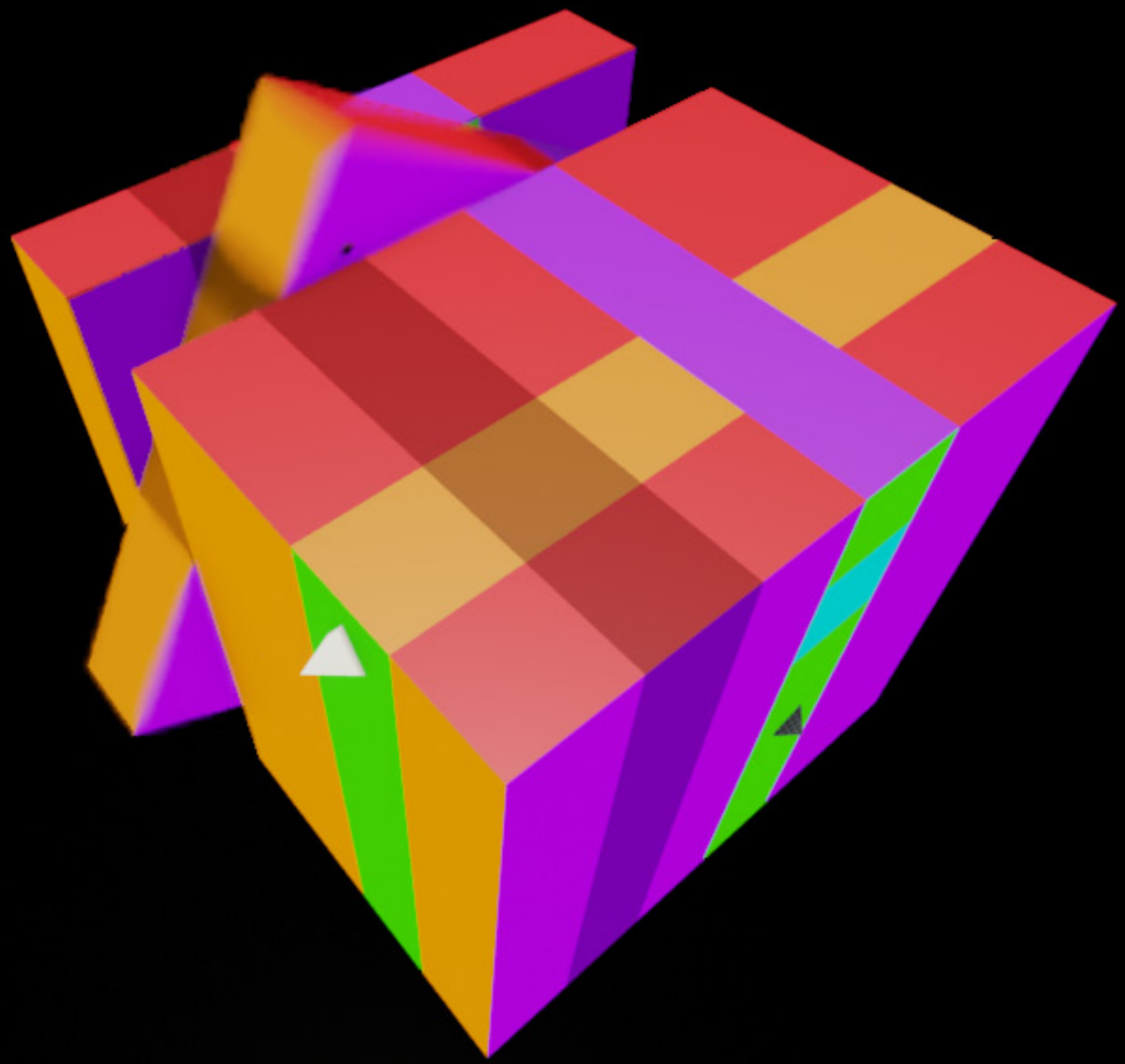
Approximate Working Hours: 12

Because we had not defined our game mechanics yet and are still about to experiment and try to figure out what kind of game to design based on the Rubik's Cube, I thought it would be handy to be able to customize and adjust the Rubik's Cube in many different ways.

In order to guarantee an easy and quick customization process, I have created a "Game Mode Blueprint", which gathers all modifiable variables for better overview and easy access.

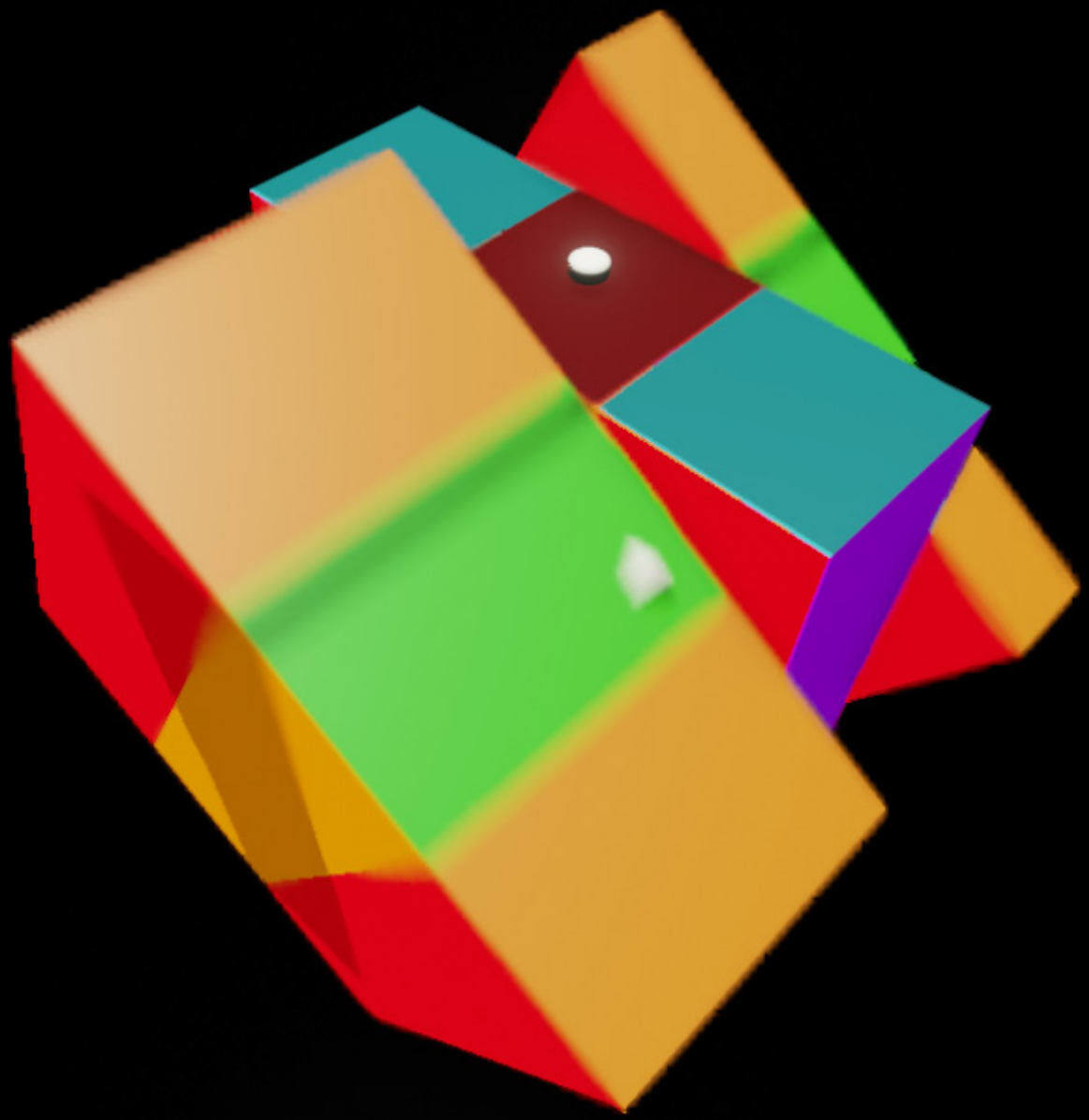
I have then implemented further modification options for the Rubik's Cube which are selectable and adjustable through this "Game Mode Blueprint".

Setting Examples



Rubik's Size: 6
Cubes Scale: 1.5
Show Marked: Rows: true

Two Row Mode: false
Inverse Rotation: false

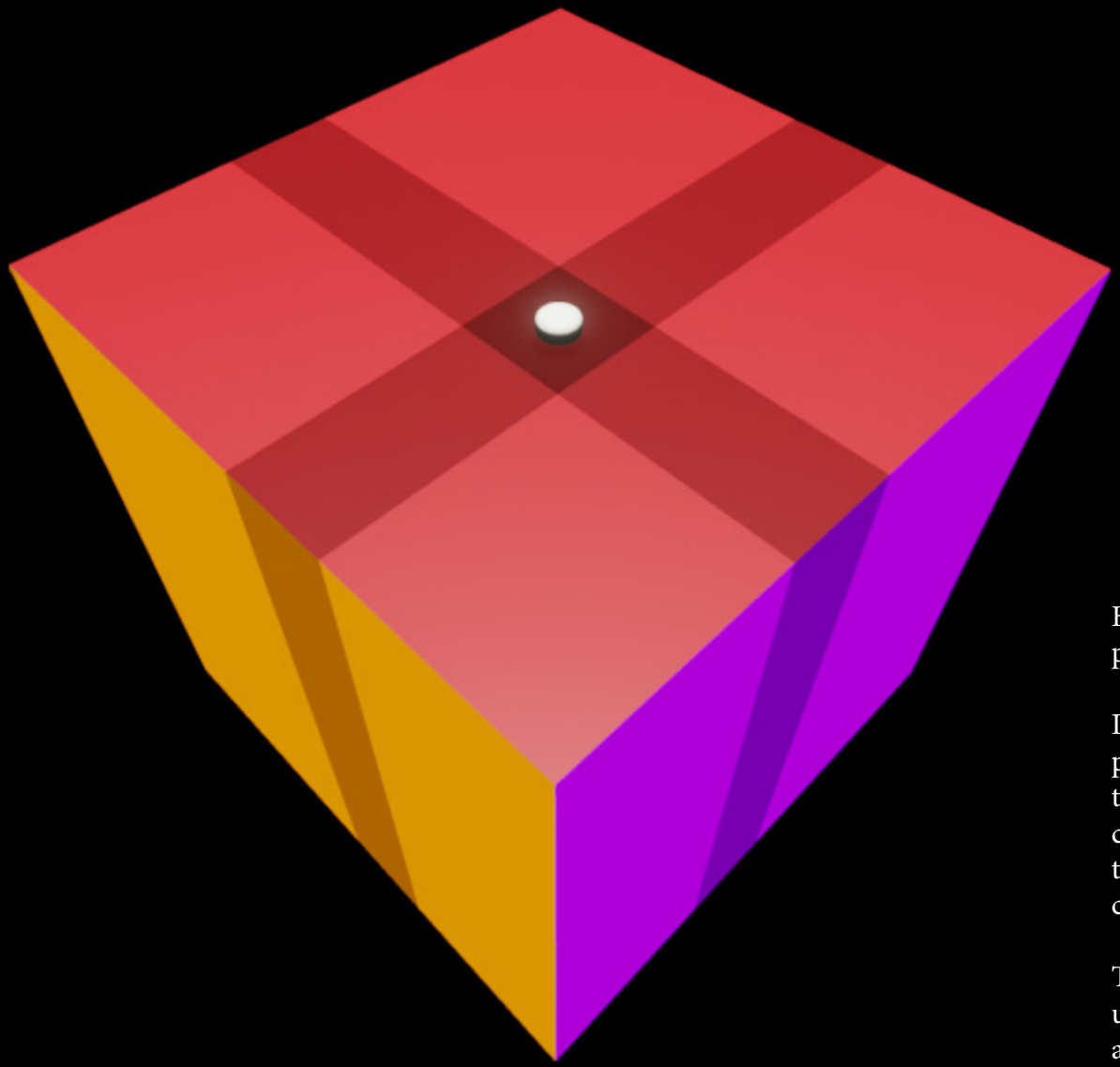


Rubik's Size: 6
Cubes Scale: 1.5
Show Marked: Rows: true

Two Row Mode: true
Inverse Rotation: true

Adding a Playable Character

Approximate Working Hours: 6



Although there was already a lot of modification options to customize the Rubik's Cube, I was still missing essential functions.

In order to place or attach any game object onto the surface of the Rubik's Cube, I decided to add "Spawn Points" on top and bottom of the rubiks cube, which would allow us to spawn any arbitrary game object for prototyping purpose.

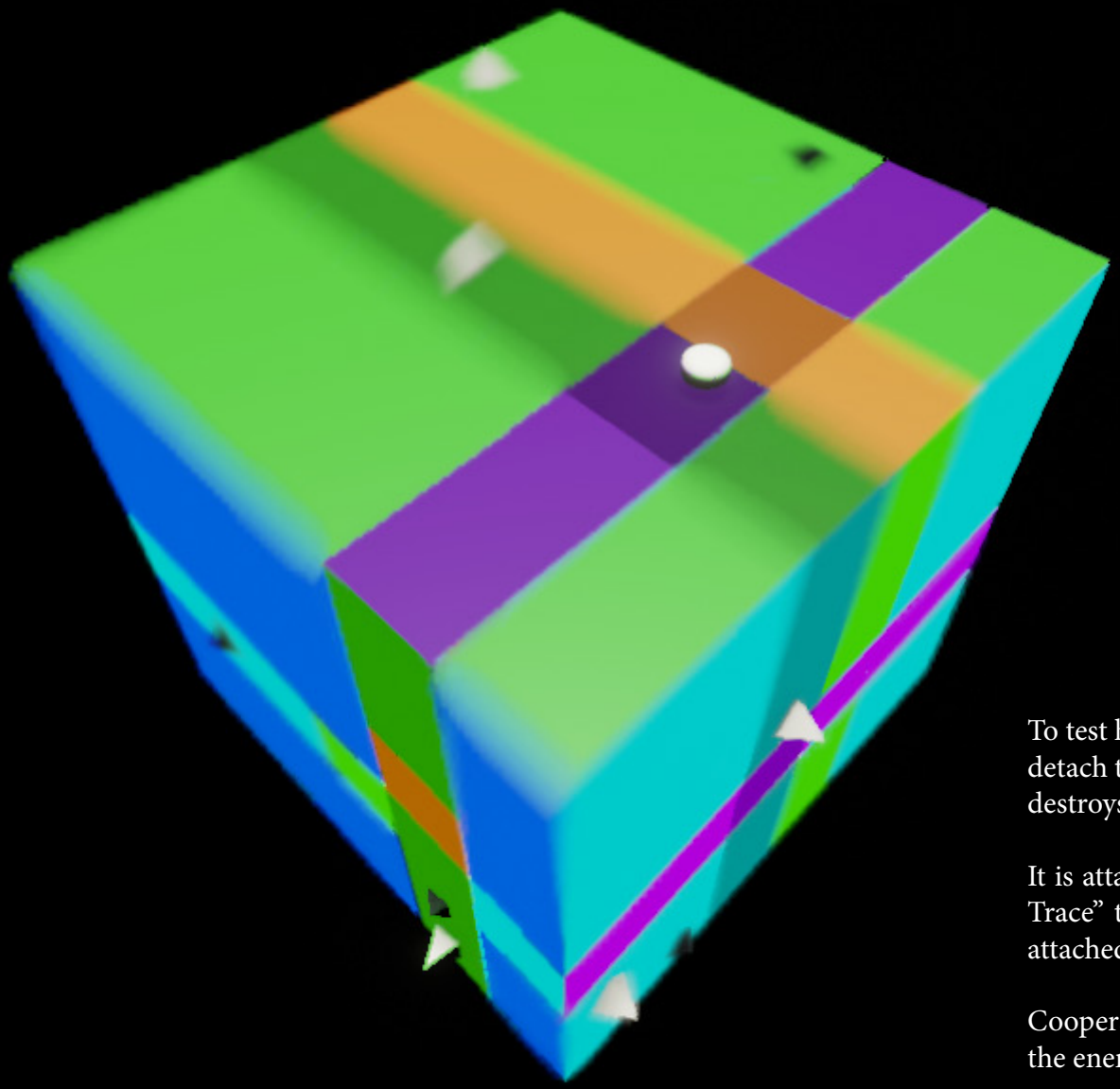
Having a functional Rubik's Cube, I decided to add a placeholder playable character for prototyping purpose.

It was important to me, that the camera remains freely placeable, so that different view angles could be quickly tested while prototyping. Because of the freely movable camera, I needed the controls of the playable character to be orientated in dependency of the orientation of the camera.

The controls for movement and rotation are now configuring themself automatically in order to keep the controls always consistent no matter the position and orientation of the camera.

Adding an Enemy

Approximate Working Hours: 2



To test how to attach game objects onto the surface and then detach them again, I have added a placeholder enemy, which destroys the playable character on overlap.

It is attaching itself to the Rubik's Cube by shooting a "Line Trace" to the direction of the Rubik's Cube and would stay attached until it gets rotated to the top surface.

Cooperatively, I have added with Ragnar and Sam a script to the enemy, which makes it slowly move towards the player.