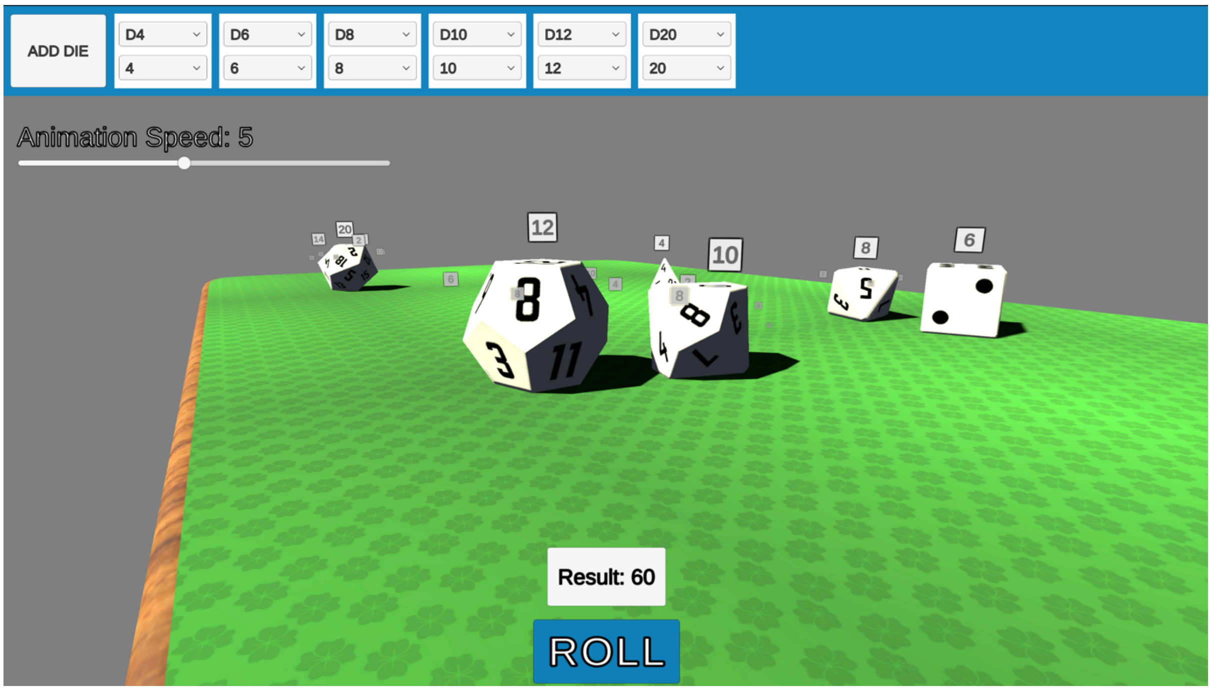
DETERMINISTIC DICE ROLLER



OBJECTIVE

The deterministic roll system allows for the quick generation and use of dice that can have pre-determined results.

This can be useful to speed up the game by allowing the player to skip the roll; allowing the designer to tweak rolls beforehand to penalise or aid the player; or ensuring server-side control of the rolls in games that need to ensure that there is no cheating going on.

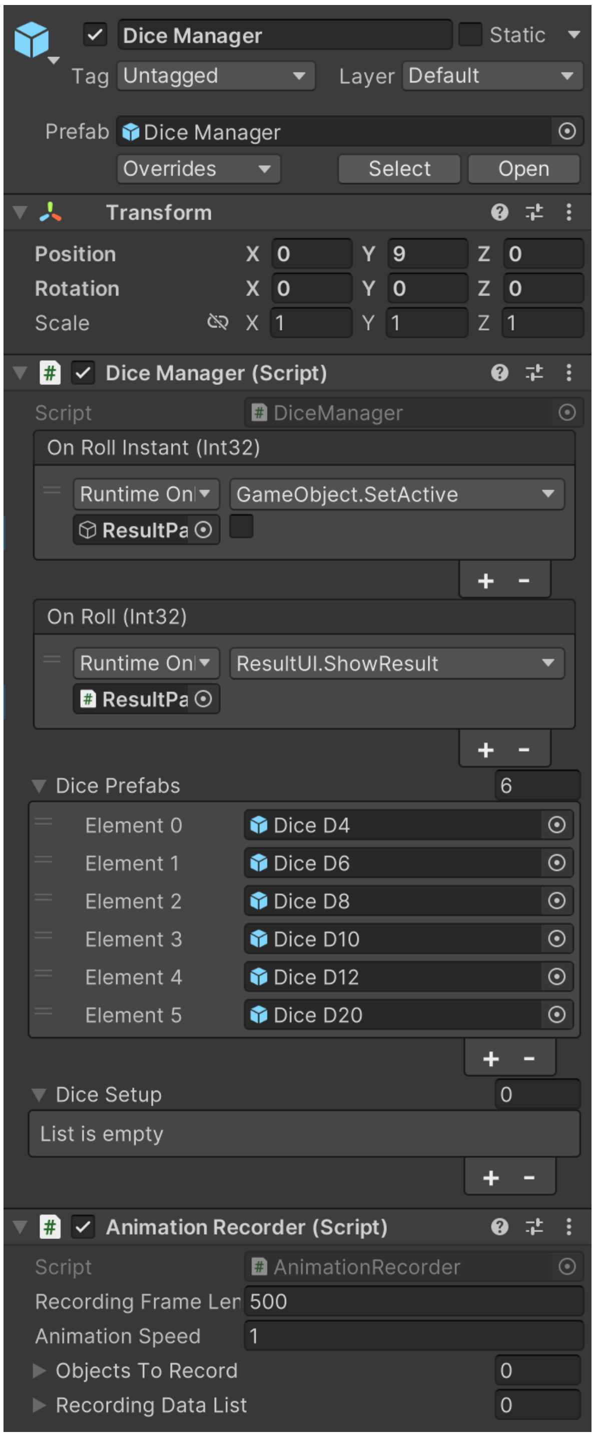
Many more uses can be found of this system to expand your design space and refine your experience.

SETUP

Just import the package and drag the **DiceManager** prefab in the scene, from here you can setup your dice throw and your event callbacks, then you are good to go.

Let's have a look at the **DiceManager** a bit more in depth:

DICE MANAGER


Dice Manager ✓
Static
Tag: Untagged
Layer: Default
Prefab: Dice Manager
Overrides | Select | Open

Transform
Position X 0 Y 9 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Dice Manager (Script) ✓
Script: DiceManager
On Roll Instant (Int32)
Runtime Onl | GameObject.SetActive
ResultPa
On Roll (Int32)
Runtime Onl | ResultUI.ShowResult
ResultPa
Dice Prefabs: 6
Element 0: Dice D4
Element 1: Dice D6
Element 2: Dice D8
Element 3: Dice D10
Element 4: Dice D12
Element 5: Dice D20
Dice Setup: 0
List is empty

Animation Recorder (Script) ✓
Script: AnimationRecorder
Recording Frame Len: 500
Animation Speed: 1
Objects To Record: 0
Recording Data List: 0

The manager contains two main components:

1. The **DiceManager** itself
2. The **AnimationRecorder**

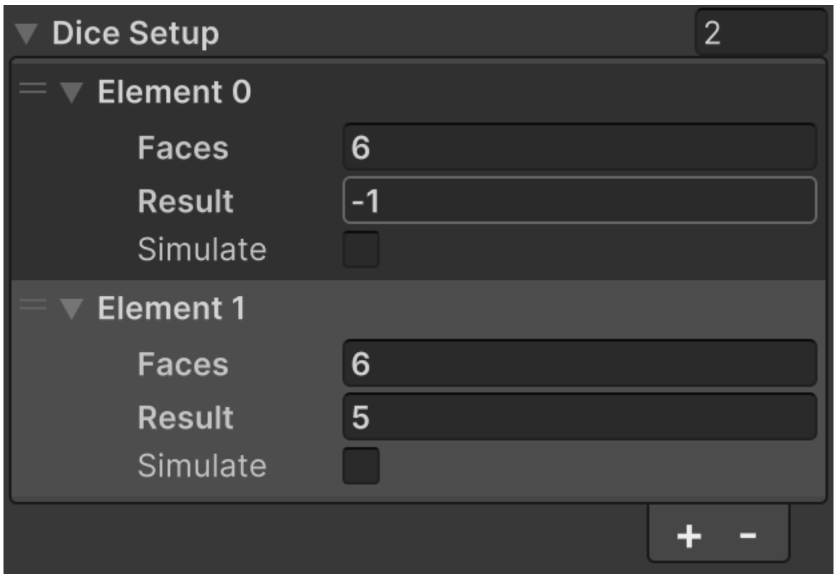
1. THE MANAGER

The manager takes care of handling the dice throw, setting up the dice to throw and fire events when the roll is determined and finished.

The Dice Prefabs are already setup for you but if you wish you can replace them, we will have a look at them later on in this document.

DICE SETUP

The Dice Setup is a section that allows to setup a set of dice manually instead of programmatically, this is what an example setup looks like:



Dice Setup 2
Element 0
Faces 6
Result -1
Simulate
Element 1
Faces 6
Result 5
Simulate

Here we see that when rolling, two **6-sided** dice are going to be thrown, the first result will be ***random*** (set to -1) and the second die will have a ***pre-determined*** result of 5.

EVENTS

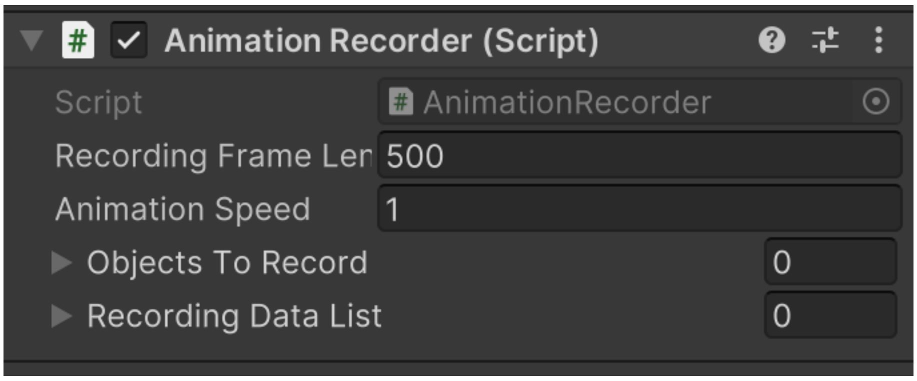
Events are triggered at two points:

1. Instantly after the physics simulation has run
2. At the end of the animation for the dice roll

This allows you to hook into the system whenever you want.

Both events pass an integer number to the connected methods, containing the ***total result*** of the dice roll.

2. THE RECORDER



Animation Recorder (Script)

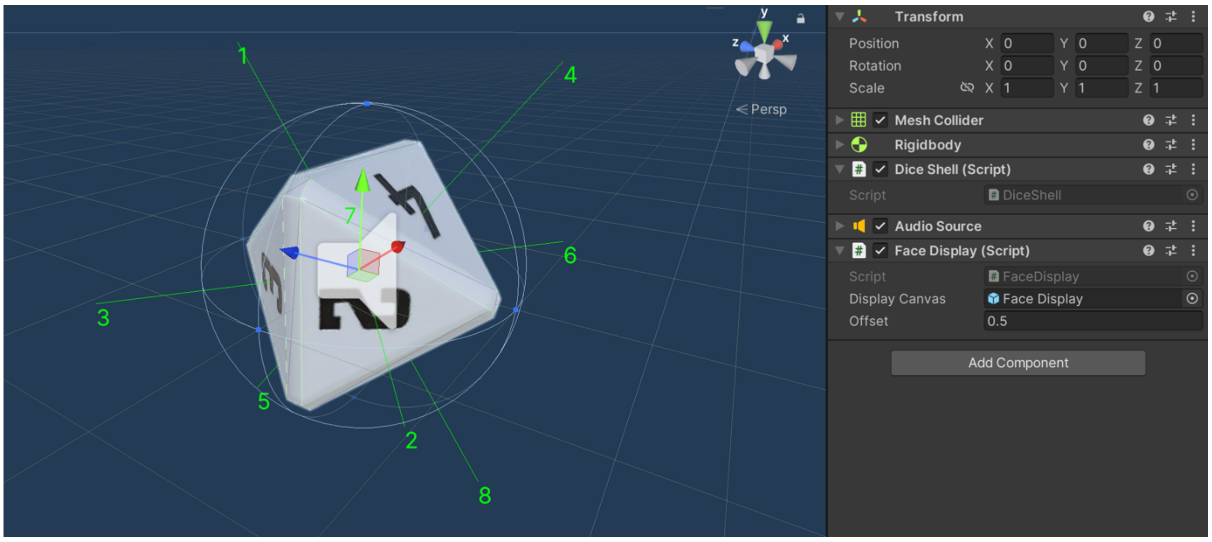
Script	AnimationRecorder
Recording Frame Len	500
Animation Speed	1
▶ Objects To Record	0
▶ Recording Data List	0

The recorder has two main settings:

The recording length that allocates frames to record for the playback animation of the dice rolling, this is stored in milliseconds, the default time is set to 5 seconds but if you see that your dice stop before they should you can easily extend this.

The animation speed is the speed at which the dice is simulated and the playback is run. This can be useful to give immediate feedback to the player while still showing the dice roll, play around with speed to find something that works for you.

DICE



Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Mesh Collider
Rigidbody
Dice Shell (Script)
Script DiceShell
Audio Source
Face Display (Script)
Script FaceDisplay
Display Canvas Face Display
Offset 0.5
Add Component

The dice have three main components:

1. The **DiceShell**
2. The **FaceDisplay**
3. The **Dice**

They all perform different functions, let's have a look at them.

THE DICE SHELL

The shell script takes care of handling events for the **AnimationRecorder** inside the physics simulation.

Additionally, it handles audio playback for when the dice collide in the animation playback.

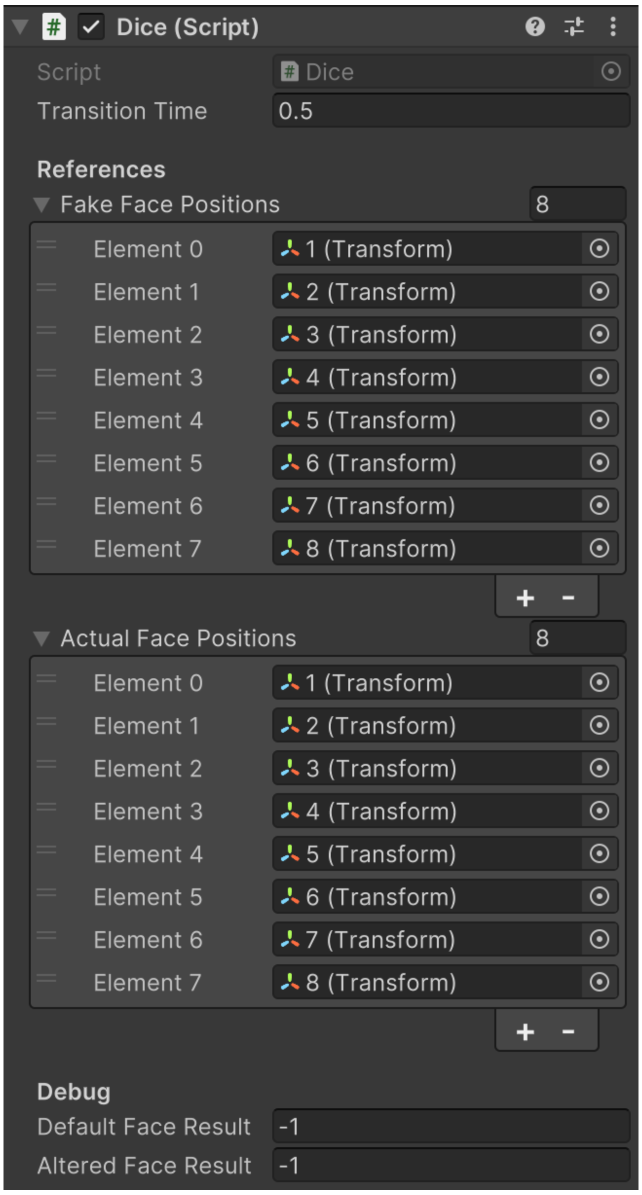
THE FACE DISPLAY

This is a utility script that spawns a ***visualization UI*** that shows the face as a number offset from the dice. This can be useful to clarify what face the dice have landed on in conditions of low light or scarce visibility.

To remove this functionality you just disable the script or remove it altogether from the prefabs.

The ***visualization UI*** prefab can also be swapped or customised from the prefab itself.

THE DICE



Dice (Script)

Script	Dice
Transition Time	0.5

References

Fake Face Positions: 8

Element 0	1 (Transform)
Element 1	2 (Transform)
Element 2	3 (Transform)
Element 3	4 (Transform)
Element 4	5 (Transform)
Element 5	6 (Transform)
Element 6	7 (Transform)
Element 7	8 (Transform)

Actual Face Positions: 8

Element 0	1 (Transform)
Element 1	2 (Transform)
Element 2	3 (Transform)
Element 3	4 (Transform)
Element 4	5 (Transform)
Element 5	6 (Transform)
Element 6	7 (Transform)
Element 7	8 (Transform)

Debug

Default Face Result	-1
Altered Face Result	-1

A D8 dice configurator

The dice component is a setup script that allows you to setup any type of dice you might want, since simulations are all physically driven you can setup any shape you wish.

The default and altered face results are driven by the **DiceManager** so there's no point in changing them.

The only configuration setting in here you might be interested in is the **Transition Time** field. This takes care of spherically interpolate the dice when a pre-determined result is set, so that the dice naturally rotates in the air. You can set it lower if you want a sharper rotation, but a transition time of half a second is enough to present an invisible and natural rotation that will make it so that the player has no idea anything happened at all.

DICE MANAGER API

The **DiceManager** can be activated and be set up by code, here's the public interface:

ROLL()

This method will roll all the dice present in the setup (see "DICE SETUP" above).

The dice not already present in the scene will be spawned at the manager's location.

It returns the total roll number of the dice.

It fires the following events:

- OnRollInstant => int result of the roll
- OnRoll => int result of the roll

RESET()

The reset method clears the animation and cleans the dice.

ADDDICE(FACE, NUMBER)

This method adds a **number** of dice with number **face** to the setup collection, it can be used to dynamically add dice whenever needed.

REMOVEDICE(FACE, NUMBER)

This method removes a **number** of dice with number **face** from the setup collection. If **number** is -1, then all the dice of the same **face** will be removed from the collection.

PAUSEDICE(FACE, NUMBER)

This method pauses a **number** of dice with number **face** in the setup collection. If **number** is -1, then all the dice of the same **face** will be paused in the collection.

It returns the dice that have been paused.

UNPAUSEDICE(FACE, NUMBER)

This method un-pauses a **number** of dice with number **face** in the setup collection. If **number** is -1, then all the dice of the same **face** will be un-paused in the collection.

It returns the dice that have been un-paused.

QUESTIONS?

If you have questions or find bugs in the asset, please contact me at commerce@gabe.software and I'll be happy to help!