# Throw Object 3D: Mobile, Desktop, FPS

Throw Object 3D (Unity Asset) — highly customizable Advanced Throwing System for Unity. You can throw object forward or in any direction with custom force, a center of mass and 100 other parameters.

Contents

1. Features of Throw Object 3D
2. Package is a Part of Unity Assets
3. Throw Object — Tons of Customizable Parameters
4. 10 customized Throwing Objects
5. Use Cases
6. Modes
    6.1. Click Or Tap
    6.2. Flick
7. Dynamic Sound System
8. Scripts
    8.1. RandomObjectPooler.cs — Object Pool
    8.2. ThrowControl.cs
    8.3. ThrowingObject.cs
9. Colliders
    9.1. Rotation (Torque) & Center of Mass
    9.2. Convex Mesh
10. Fading (Dissolving) VFX
    10.1. Shaders
11. Tutorial
12. Testing
    12.1. Tested with Devices
13. Support
14. Changelog

## Features of Throw Object 3D

All modules are designed independently to keep this asset extendable & easy to understand:

- ⭐ Cross-platform: Mobile, Desktop (Mouse);
- ⭐ 2 Throwing Modes: “Flick” & “Click Or Tap”;
- ⭐ 2 Demos: Mobile, Desktop (FPS);
- ⭐ 7 Events;
- ⭐ 10 customized Throwing Objects;
- ⭐ Easy implementation of your own Throwing Objects;
- ⭐ Tons of Customizable Parameters;
- ⭐ Dynamic Sound System (depends on Speed of Throwing Object);
- ⭐ Layer Changing (actual for quick Throwing to neutralize mutual collisions);
- ⭐ Fading In & Fading Out: Dissolving — Material Control in Runtime;
- ⭐ Optimizations: Object Pool.

## Package is a Part of Unity Assets

Basketball Game (docs).

---

AR Basketball GO (docs).

---

AR Toss Boss (docs).

---

AR Throwing (docs).

## Throw Object — Tons of Customizable Parameters

> *Input sensitivity, force, torque, delays, position, rotation, fading, etc.:*

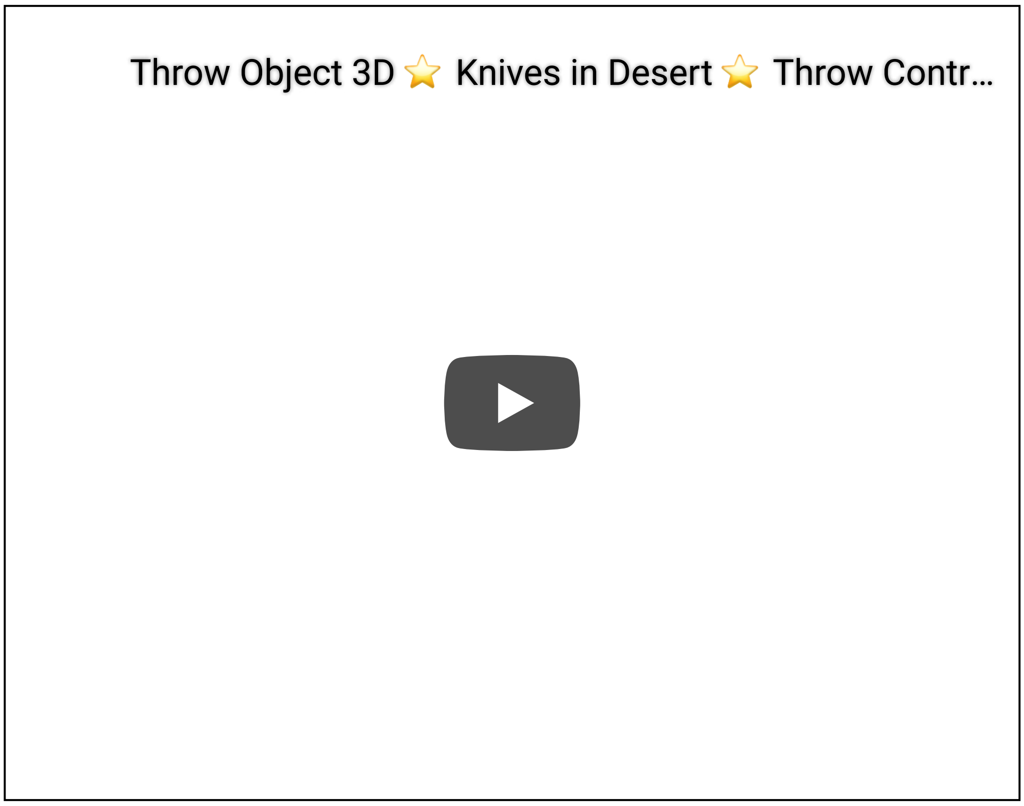
*— For all Throwing Objects at once;*
*— For each Throwing Objects individually.*

## 10 customized Throwing Objects

- ⭐ Weapons: Axe, Sword, Broken Sword,
- ⭐ Balls: Basketball, Brick Ball (Grey), Brick Ball (Red);
- ⭐ Miscellaneous: Coin, Shovel, Horseshoe, Tree (branch).

## Use Cases

- ⭐ A game in the style of Pokemon GO (iOS, Android),
- ⭐ A game in the style of Can Knockdown (iOS, Android),
- ⭐ Paper Toss Boss;
- ⭐ Paper Bin AR;
- ⭐ FPS with Throwing Objects;
- ⭐ AR, VR games with Throwing Objects;
- ⭐ Survival Game;
- ⭐ Any of your Bold Fantasies.



## Modes

The higher the click (or tap) position on the screen, the greater the strength of the throw.

The system completely works on desktop as well as on mobiles: you can throw object with the mouse or with a finger.

## Click Or Tap

In this mode, the Game Object is throwing in the direction of the click (desktop) or tap (mobile).

### FPS

For the FPS game, there is an option for fixing the input position in a specific place on the screen (by default in the center of the screen). Desktop Demo uses CharacterController (Throw force takes into account the speed of the player's movement).

## Flick

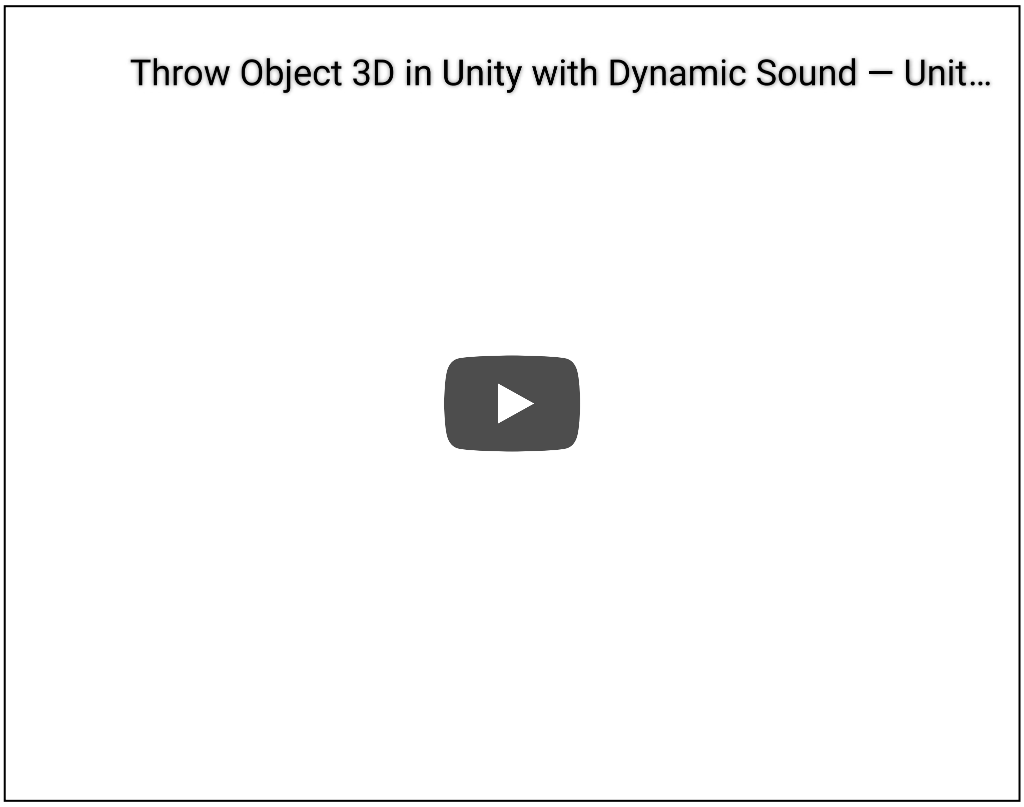
In this mode, you can throw object in the style of the Pokemon GO game (iOS, Android) (by flicking it from the bottom of the screen up toward the target).

### Is Full Path For Flick?

If it's false then it allows fast flicks only: positions in the last and previous frames are taken into account.

## Dynamic Sound System

Sound depends on the Speed of Throwing Object.



## Scripts

Throwing System consists of 3 control scripts.

### *RandomObjectPooler.cs* — Object Pool

Object Pool is the performance optimization technique through Objects Reusing. All Throwing Objects are created on start and used throughout the game session.

### *ThrowControl.cs*

This is the main script that manages throw & all Throwing Objects on a scene through Object Pool.
The script uses disabling and enabling the following nodes: Colliders, Triggers, Renderers.

#### Work Scheme

1. Initialize Object Pool,
2. Initialize Throwing Objects,
3. Get the First Throw,
4. Get Next Throw.

### Events

- ⭐ Next Events have *ThrowingObject* as a parameter. So you have more flexibility with access to instances of *ThrowingObject.cs* in outer functions assigned to Events.
    - ⭐ OnThrow,
    - ⭐ OnNextThrowGetting,
    - ⭐ OnReset,
    - ⭐ OnFadingOut,
- ⭐ OnInitialized.

### Coroutines

The only script that uses coroutine calls.

Because of coroutines behavior, it is impossible to place some methods inside Throwing Object script.
OOP does not work in this case, but we have a stable throwing with Object Pool.

## *ThrowingObject.cs*

The script contains basic functions and parameters to operate Throwing Object.
This script must be attached to each Throwing Prefab (check tutorial).

### Events

- ⭐ OnThrow;
- ⭐ OnResetPhysicsBase.

### Custom Data

- ⭐ Mono Behaviour Custom: it is useful to assign a specific control script for a unique type of Throwing Object and access to it outside the Throwing System. Example: *BasketballBallControl.cs* in AR Basketball GO (docs) & Basketball Game (docs).
- ⭐ Flag Custom: for Any User Logic outside the Throwing System.

## Colliders

### Rotation (Torque) & Center of Mass

There are some nuances with Rigidbody.centerOfMass when rotation.

#### Symmetric Mesh

If you use Mesh Collider for Throwing Game Object and your mesh is ideally symmetric then it is the perfect case and game object will throw naturally.

#### Asymmetric Mesh

If you use Mesh Collider for Throwing Game Object and your mesh is asymmetric then you will have unnatural throw behavior when rotation.

##### Solution 1: Custom Center of Mass

You can indicate Custom Center of Mass manually in the Editor.
Unity Package contains some examples with Custom Center of Mass.

There is no magic function that calculates the right Center of Mass.

##### Solution 2: Simple forms of colliders

You need to use one or more simple forms of colliders (box, sphere, etc.) placing them on the same Throwing Game Object and along the straight line. So in this case, we have the right center of mass & right rotation.

### Convex Mesh

Unity allows you to use only convex mesh colliders. In some cases, it is not what you need because the collider covers more space, than the object itself in some points.

In order to quickly & automatically get a convex mesh collider for complex objects, use Technie Collider Creator.

# Fading (Dissolving) VFX

It's Separate Independent Script for Throwing Objects: *MaterialControl.cs* (It's used by *ThrowingObject.cs*). You can use it for Fading outside Throwing System (Examples: Ring, Ring Holder, Net, Backboard, Pole in AR Basketball GO (docs) & Basketball Game (docs)).

Here you can control Fading In & Fading Out with Full Control of Value, Delay, Time & Speed for Indicated Shader Parameter.

## Shaders

There are 3 Custom Shaders with Shadow Fading & "Slice Guide" Texture for Fading (Your Material > Shader > Custom > Makaka Games):

- ⭐ Diffuse (Dissolving),
- ⭐ Bumped Diffuse (Dissolving),
- ⭐ Particles Cutout Transparent — for Transparent Textures (Example: Net in AR Basketball GO (docs) & Basketball Game (docs)).

# Tutorial

> *This tutorial is relevant for 4.1+ version of Asset. Tutorials for previous versions can be found in the asset folder.*

## Getting Started with Throw Object 3D

Folders in the package by Default:

- ⭐ Makaka Games.

### Steps

*If you have any issues with the first launch then just Reach Support with Invoice Number and Get Help.*

*If you read this tutorial from PDF, first check the latest docs online to get actual information.*

1. Create New Unity Project with Unity 2021.1.6.
2. File > Build Settings > Switch to Target Platform.
3. Download and import Throw Object 3D into Unity.
4. Reopen Unity Project.
5. Open Scene: Makaka Games > Throw Object 3D > Scenes > Demo.
6. Test In Unity Editor or Build for Mobile or Desktop.

## Object Count

1. Hierarchy View > "ThrowingPoolControl" Game Object.
2. Inspector View > "Random Object Pooler" script > "Init Pooled Amount" Parameter.

## Creating your own Throwing Object Prefab

1. Create Prefab with a template:
    1. Drag one of the customized throwing prefabs into the scene,
    2. Name your Game Object,
    3. Drag named Game Object into Project View to create your own Prefab;
2. Select Game Object on the Scene & Customize it:
    1. Indicate Mesh in Mesh Filter component,
    2. Customize Material & its Fading (see "Fading" chapter) in Mesh Renderer component (Duplicate Material in Project View and indicate here to separate different Throwing Objects),
    3. Customize Colliders:
        1. Indicate Mesh in Mesh Collider component (or in more suitable collider component);
    4. Option: Customize the Center of Mass for more natural Throwing;
3. Customize the Transform component as you wish;
4. Apply Changes to Prefab;
5. Delete Game Object from Scene;
6. Option: Add Asset Label for Prefab to convenient searching in Project View;

7 Add Prefab to *RandomObjectPooler* component of *ThrowingPoolControl* Game Object;

8 Throw Object Forward in “Play Mode”.

## Change Spawn Position of Throwing Objects

See the “Position” section in the Editor of *ThrowingObject.cs* for each Throwing Prefab individually.

- ⭐ [X, Y]: “Position In Viewport On Reset”,
- ⭐ [Z]: “Camera Near Clip Plane Factor On Reset”.

## Change the Color of Throwing Objects in Runtime

*SetMaterial()* function is used instead of *SetColor()* functions all over the system to avoid memory allocations.

So if you plan to change Colors of Throwing Objects in Runtime outside Throwing System you need to create Materials with Target colors in advance.

Example: Red Fail Material in AR Basketball GO (docs) & Basketball Game (docs).

## Testing

Read Article: Mobile Testing.

## Tested with Devices

- ⭐ iOS on iPhone XS Max;
- ⭐ Android on Samsung Galaxy A71;
- ⭐ macOS X;
- ⭐ Windows 11.

## Support

First, read the latest docs online.
If it didn’t help, get the support.

## Changelog

*Check the current version on Asset Store.*
*The latest versions will be added as soon as possible.*

4.1:

- ⭐ Unity 2021.1.6

4.0:

This version adds performance improvements by more effective handling of materials, shaders, fading & caching, so it’s incompatible with previous versions of Asset.

- ⭐ Improve Dissolving (Fading) VFX. Separate Independent Script for Throwing Objects: *MaterialControl.cs* — so you can use it for Fading outside Throwing System (Examples: Ring, Ring Holder, Net, Backboard, Pole in AR Basketball GO (docs) & Basketball Game (docs)).
    - ⭐ 1 Material Instead of 2: No Memory Allocation while Fading.
    - ⭐ 3 Custom Shaders with Shadow Fading & "Slice Guide" Texture for Fading:
        - ⭐ Diffuse,
        - ⭐ Bumped Diffuse,
        - ⭐ Particles Cutout Transparent — for Transparent Textures (Example: Net in AR Basketball GO (docs) & Basketball Game (docs)).
    - ⭐ Fading In & Fading Out with Full Control of Value, Delay, Time & Speed for Indicated Shader Parameter.
- ⭐ Add public event Actions for Each Trowing Object in *ThrowingObject.cs*:
    - ⭐ OnThrow;
    - ⭐ OnResetPhysicsBase.
- ⭐ Improve all public Events in *ThowControl.cs*: add ThrowingObject as a parameter. So you have more flexibility with access to instances of *ThrowingObject.cs* in outer functions assigned to Events.
- ⭐ Add a Custom MonoBehaviour field in *ThrowingObject.cs:* it is useful to assign a specific control script for a unique type of Throwing Object and access to it outside the Throwing System. Example: *BasketballBallControl.cs* in AR Basketball GO (docs) & Basketball Game (docs).
- ⭐ Now *SetMaterial()* function is used instead of *SetColor()* functions all over the system to avoid memory allocations. So if you plan to change Colors of Throwing Objects in Runtime outside Throwing System you need to create Materials with Target colors in advance. Example: Red Fail Material in AR Basketball GO (docs) & Basketball Game (docs).
- ⭐ Unity 2019.3:
    - ⭐ Fix Mesh Collider of Horseshoe.

3.17:

- ⭐ “SoundSettings” class and “PlayRandomSoundDependingOnSpeed(ThrowingObject.SoundSettings audioSettings, GameObject to)” function. Now you can easily declare Sound Settings in your own scripts to Play Sounds based on the speed of Throwing Objects. It is helpful when you implement a dynamic sound system when Throwing Objects collide with Floor, Walls and Other Game Objects.
- ⭐ Function: public int GetObjectCount ();
- ⭐ Function: public void SetColor (Color color, GameObject to);

3.16 (Bug Fixing):

- ⭐ Audio Delay (Latency) on Mobiles;
- ⭐ “Index Out of Bounds” error in Input.GetTouch(0) for Mobiles;
- ⭐ Debug.LogWarning() for unknown platforms.

3.15:

- ⭐ Option: Custom Flag in ThrowingObject.cs for Any User Logic.

3.14:

- ⭐ Option: Tag Selector for ThrowControl.cs to set a custom tag for all objects in the pool.

3.13:

- ⭐ Unity 2019.1;

3.12:

- ⭐ Unity 2018.3;

3.7:

- ⭐ Move cameraNearClipPlaneFactor to ThrowingObject.cs to customize the whole position (X, Y, Z) of Throwing Object prefab in one place;
- ⭐ Fix enabling and disabling colliders for compound Throwing Prefabs.

3.6:

- ⭐ Unity 2018.2;
- ⭐ Fix errors: When Flick Mode you just click on the object without finger position change.

3.4 (Layer Changing in Throw Control):

Actual for quick Throwing to neutralize mutual collisions.

- ⭐ Customizations:
    - ⭐ ON / OFF,
    - ⭐ Layer Mask on Throw,
    - ⭐ Layer Mask on Reset,
    - ⭐ Delay.

3.3 (Throw Customizations in Throw Control):

- ⭐ Input Sensitivity,
- ⭐ Force Factor Extra,
- ⭐ Torque Factor Extra,
- ⭐ Torque Angle Extra,
- ⭐ Parent on Throw.

3.2 (“Flick” Mode Customizations in Throw Control):

- ⭐ Option: Is Full Path:
    - ⭐ If it’s false then it allows fast flicks only (positions in the last and previous frames are taken into account);
- ⭐ Lerp Time Factor On Touch.

3.1:

- FPS Demo integrated with CharacterController (Desktop);
- Throw force takes into account the speed of the player's movement;
- Fixing an input position in a specific place on the screen (by default in the center of the screen).

3.0 (New Architecture: Throw Control):

Main Control Script (Throw Control) operates Object Pool & All Throwing Objects now (Throwing Object Script is attached to each Throwing Object prefab).

Work Scheme:

1. Initialize Pool,
2. Initialize Throwing Objects,
3. Get the First Throw,
4. Get Next Throw.

- Customizations in Throw Control:
    - Throw Mode;
    - Events with Delays:
        - OnInitialized,
        - OnThrow,
        - OnNextThrowGetting,
        - OnReset,
        - OnFadeOut.

2.6 (Rotation Customizations):

- ⭐ Option: Rotate Object in Throw Direction;
- ⭐ Rotation on Reset:
  - ⭐ Default,
  - ⭐ Last,
  - ⭐ Random,
  - ⭐ Custom.

2.5:

- ⭐ Position in Viewport On Reset.

2.4 (Center of Mass Customizations):

- ⭐ Custom Center of Mass,
- ⭐ Logging of Center of Mass by Default.

2.3 (Torque Customizations):

- ⭐ Torque Axis,
- ⭐ Torque Angle,
- ⭐ Torque Factor,
- ⭐ Max Angular Velocity at Awake.

2.2 (Force Customization):

- ⭐ Force Factor,
- ⭐ Force Direction Extra.

2.1:

- ⭐ “Click or Tap” Throwing Mode  for Desktop & Mobiles;
- ⭐ Customization:
  - ⭐ Throwing Mode.

2.0 (Object Pool — Throwing of Multiple Objects):

- ⭐ Prefabs Using:
    - ⭐ Single (actual for Testing target prefab; None => Multiple),
    - ⭐ Multiple;
- ⭐ Generation order for Multiple Prefabs:
    - ⭐ In random order,
    - ⭐ In the right order;
- ⭐ Customizations:
    - ⭐ Pool Parent,
    - ⭐ Init pooled amount,
    - ⭐ Event (OnInitialized).

1.3 (Dynamic Sound System):

Sound System depending on Speed of Throwing Object.

- ⭐ Customizations:
    - ⭐ Array for randomly selecting Sounds,
    - ⭐ Speed Clamping,
    - ⭐ Pitch (Minimum & Factor),
    - ⭐ Volume Factor.

1.2 (Fade Out option for Thrown Objects):

- ⭐ Customizations:
    - ⭐ ON / OFF,
    - ⭐ Delay,
    - ⭐ Speed,
    - ⭐ Materials,
    - ⭐ Event (OnInitialized).

1.1:

- ⭐ Desktop version: throw object with the mouse. Use it for Standalone apps or for testing of mobile apps in Unity Editor.

1.0 (“Flick” Throwing Mode for Mobiles):

⭐ Throwing of Single Object in “Flick” Mode for Mobiles (AR Basketball GO — Unity Asset).