Throw Object 3D by makaka.org

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 80+ other throwing parameters. 2 Modes. 2 Demos. 10 Throwing Objects.

Contents

Features of Throw Object 3D

All modules are designed independently to keep the Unity Asset extendable & easy to understand:

- ⭐ Cross-platform: Mobile, Desktop (Mouse).
- ⭐ 2 Throwing Modes: "Click Or Tap" (Easy) and "Flick" (or Swipe: Hard).
- ⭐ 80+ Customizable Parameters — sensitivity, force, torque, delays, position, rotation, fading, etc.:
  - ⭐ for all Throwing Objects at once;
  - ⭐ for each Throwing Object individually.
- ⭐ 2 Demos: Mobile, Desktop (FPS).
- ⭐ 7 Events.
- ⭐ 10 customized Throwing Objects:
  - ⭐ Weapons: Axe, Sword, Broken Sword;
  - ⭐ Balls: Basketball, Brick Balls (Grey, Red);
  - ⭐ Miscellaneous: Coin, Shovel, Horseshoe, Tree/Branch.
- ⭐ Easy implementation of your own Throwing Objects.
- ⭐ Dynamic Sound System — play Sounds based on speed, pitch, and volume factors of Throwing Objects upon:
  - ⭐ Throw,
  - ⭐ Any Custom Game Logic (e.g., Collisions with Different Surfaces). Example: AR Throw & Score (docs).
- ⭐ Dynamic Material System:
  - ⭐ Fading In & Fading Out: Dissolving,
  - ⭐ Material Changing in Runtime for Any Custom Game Logic (e.g., fail, win). Example: AR Throw & Score (docs).
- ⭐ Layer Changing (to avoid unwanted collisions).
- ⭐ Optimizations:
  - ⭐ Object Pool for Throwing Objects to manage the memory;
  - ⭐ TextMesh PRO for Texts to update them when really needed.

Package Contains

- ⭐ Demo Scene with Throwing for Desktop & Mobiles with Static Camera.
- ⭐ Demo Scene with Throwing for Desktop (FPS): Walk, Run, Jump, Throw.
- ⭐ Menu Scene.
- ⭐ Loading Screen to switch scenes seamlessly.

Check the Map of Unity Assets to choose the product that best suits your needs.

Package is a Part of Unity Assets

Basketball Game (docs).

AR Throw & Score (docs).

AR Throwing (docs).

AR Basketball GO (docs).

Use Cases

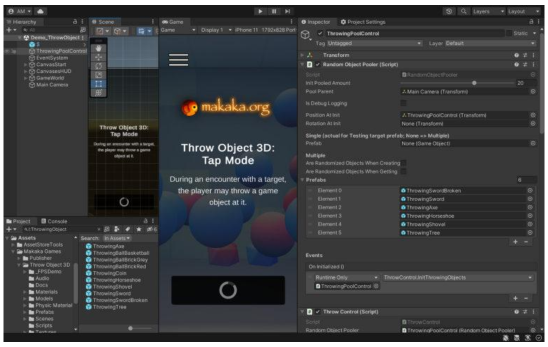
- ⭐ Games in the style of Pokemon GO, Can Knockdown, Paper Toss Boss, Paper Bin AR;
- ⭐ AR Unity Assets by Makaka Games;
- ⭐ FPS, VR Games with Throwing Objects;
- ⭐ Survival Games;
- ⭐ Any of your Bold Fantasies.

Throwing System: 80+ Customizable Parameters

> *Throwing System consists of 4 control scripts that are completely customized already in Demo Scenes.*

1. Random Object Pooler

RandomObjectPooler.cs — independent script: Object Pool, performance optimization technique through Objects Reusing. It's used by *ThrowControl.cs*. All Throwing Objects are created at the Scene Start and used throughout the Scene Session.


makaka.org
Throw Object 3D:
Tap Mode
During an encounter with a target, the player may throw a game object at it.

```
ThrowingPoolControl                         Static
Tag Untagged                  Layer Default
Transform
Random Object Pooler (Script)
Script                     RandomObjectPooler
Init Pooled Amount         20
Pool Parent                Main Camera (Transform)
Is Debug Logging
Position At Init           ThrowingPoolControl (Transform)
Rotation At Init           None (Transform)
Single (actual for Testing target prefab; None => Multiple)
Prefab                     None (Game Object)
Multiple
Are Randomized Objects When Creating
Are Randomized Objects When Getting
Prefabs                    6
  Element 0                ThrowingSwordBroken
  Element 1                ThrowingSword
  Element 2                ThrowingAxe
  Element 3                ThrowingHorseshoe
  Element 4                ThrowingShovel
  Element 5                ThrowingTree
Events
On Initialized ()
  Runtime Only             ThrowControl.InitThrowingObjects
  ThrowingPoolControl
Throw Control (Script)
Script                     ThrowControl
Random Object Pooler       ThrowingPoolControl (Random Object Pooler)
```

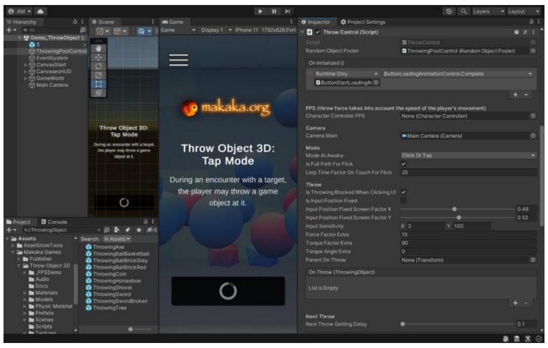
Parameters

- ⭐ *initPooledAmount* — to regulate Object Count in the Scene at Start.
- ⭐ *areRandomizedObjectsWhenCreating* – when you need a random instance count of possible Throwing Object prefabs for every Scene Start.
- ⭐ *areRandomizedObjectsWhenGetting* – when you require a random Throwing Object instance for every Throw.

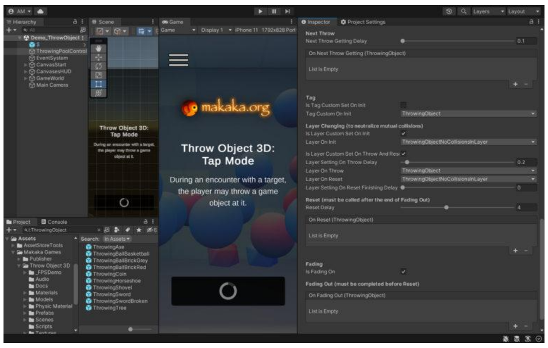
2. Throw Control

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

To Get the Count of Throwing Objects, use *GetObjectCount()* function.


makaka.org
Throw Object 3D:
Tap Mode
During an encounter with a target, the player may throw a game object at it.

```
Throw Control (Script)
Script                                   ThrowControl
Random Object Pooler                     ThrowingPoolControl (Random Object Pooler)
On Initialized ()
  Runtime Only   ButtonLoadingAnimationControl.Complete
  ButtonStartLoadingAr
FPS (throw force takes into account the speed of the player's movement)
Character Controller FPS                 None (Character Controller)
Camera
Camera Main                              Main Camera (Camera)
Mode
Mode At Awake                            Click Or Tap
Is Full Path For Flick                   ✓
Lerp Time Factor On Touch For Flick      20
Throw
Is Throwing Blocked When Clicking UI     ✓
Is Input Position Fixed
Input Position Fixed Screen Factor X     0.48
Input Position Fixed Screen Factor Y     0.52
Input Sensitivity                        X 2   Y 100
Force Factor Extra                       15
Torque Factor Extra                      60
Torque Angle Extra                       0
Parent On Throw                          None (Transform)
On Throw (ThrowingObject)
  List is Empty
Next Throw
Next Throw Getting Delay                 0.1
```


makaka.org
Throw Object 3D:
Tap Mode
During an encounter with a target, the player may throw a game object at it.

```
Next Throw
Next Throw Getting Delay                 0.1
On Next Throw Getting (ThrowingObject)
  List is Empty
Tag
Is Tag Custom Set On Init
Tag Custom On Init                       ThrowingObject
Layer Changing (to neutralize mutual collisions)
Is Layer Custom Set On Init              ✓
Layer On Init                            ThrowingObjectNoCollisionsInLayer
Is Layer Custom Set On Throw And Res     ✓
Layer Setting On Throw Delay             0.2
Layer On Throw                           ThrowingObject
Layer On Reset                           ThrowingObjectNoCollisionsInLayer
Layer Setting On Reset Finishing Delay   0
Reset (must be called after the end of Fading Out)
Reset Delay                              4
On Reset (ThrowingObject)
  List is Empty
Fading
Is Fading On                             ✓
Fading Out (must be completed before Reset)
On Fading Out (ThrowingObject)
  List is Empty
```

Modes

The higher the last position of the mouse cursor (or finger) during the click (or tap), the greater the strength of the throw. Throwing System completely works on Desktop as well as on Mobiles.

Click Or Tap

In this mode, the Game Object is thrown 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). FPS 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.

Work Scheme

1. Initialize Object Pool,
2. Initialize Throwing Objects,
3. Get First Throw (Start the Throwing Cycle):
    1. Get Next Throw,
    2. Fade In,
    3. Throw,
    4. Fade Out,
    5. Reset.

Events

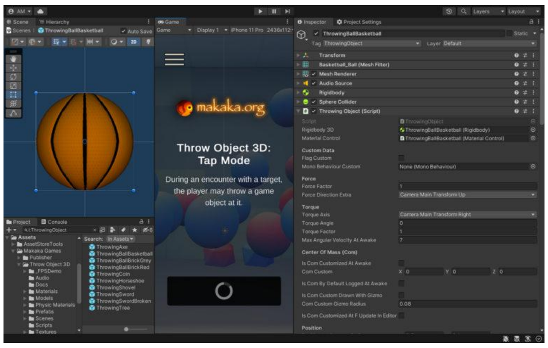
- *OnInitialized*;
- Next Events have *ThrowingObject* as a parameter. So you have more flexibility with access to instances of *ThrowingObject.cs* in outer functions assigned to Events.
    - *OnNextThrowGetting*,
    - *OnThrow*,
    - *OnFadingOut*,
    - *OnReset*.

Coroutines

The only script that starts coroutines. Because of coroutines behavior, it's impossible to place some methods inside *ThrowingObject.cs*. OOP doesn't work in this case, but we have a stable throwing with Object Pool.

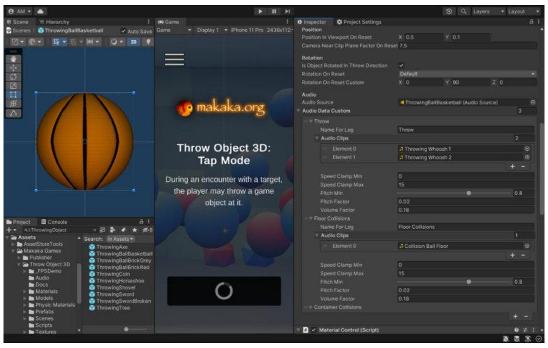
3. Throwing Object

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



makaka.org
Throw Object 3D: Tap Mode
During an encounter with a target, the player may throw a game object at it.
Inspector
Project Settings
Layers
Layout

```
ThrowingBallBasketball                         Static
Tag ThrowingObject        Layer Default
Transform
Basketball_Ball (Mesh Filter)
Mesh Renderer
Audio Source
Rigidbody
Sphere Collider
Throwing Object (Script)
Script                     ThrowingObject
Rigidbody 3D               ThrowingBallBasketball (Rigidbody)
Material Control           ThrowingBallBasketball (Material Control)
Custom Data
Flag Custom
Mono Behaviour Custom      None (Mono Behaviour)
Force
Force Factor               1
Force Direction Extra      Camera Main Transform Up
Torque
Torque Axis                Camera Main Transform Right
Torque Angle               0
Torque Factor              1
Max Angular Velocity At Awake  7
Center Of Mass (Com)
Is Com Customized At Awake
Com Custom                 X 0   Y 0   Z 0
Is Com By Default Logged At Awake
Is Com Custom Drawn With Gizmo
Com Custom Gizmo Radius    0.08
Is Com Customized At F Update In Editor
Position
```



makaka.org
Throw Object 3D: Tap Mode
During an encounter with a target, the player may throw a game object at it.
Inspector
Project Settings
Layers
Layout

```
Position
Position In Viewport On Reset          X 0.5   Y 0.1
Camera Near Clip Plane Factor On Reset 7.5
Rotation
Is Object Rotated In Throw Direction   ✓
Rotation On Reset                      Default
Rotation On Reset Custom               X 0   Y 90   Z 0
Audio
Audio Source                           ThrowingBallBasketball (Audio Source)
Audio Data Custom                      3
  Throw
    Name For Log                       Throw
    Audio Clips                        2
      Element 0                        Throwing Whoosh 1
      Element 1                        Throwing Whoosh 2
    Speed Clamp Min                    0
    Speed Clamp Max                    15
    Pitch Min                          0.8
    Pitch Factor                       0.02
    Volume Factor                      0.18
  Floor Collisions
    Name For Log                       Floor Collisions
    Audio Clips                        1
      Element 0                        Collision Ball Floor
    Speed Clamp Min                    0
    Speed Clamp Max                    15
    Pitch Min                          0.8
    Pitch Factor                       0.02
    Volume Factor                      0.18
  Container Collisions
Material Control (Script)
```

Events

- ⭐ *OnThrow*;
- ⭐ *OnResetPhysicsBase*.

Parameters

- ⭐ *flagCustom*: for Any User Logic outside the Throwing System.
- ⭐ *monoBehaviourCustom*: it's 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).
- ⭐ *positionInViewportOnReset*: spawn position [x, y].
- ⭐ *cameraNearClipPlaneFactorOnReset*: spawn position [z].
- ⭐ *audioDataCustom*: array for customizing any dynamic Audio Data based on speed, pitch, and volume factors of Throwing Object. Access from *ThrowControl.cs* or *ThrowingObject.cs* with *PlayRandomSoundDependingOnSpeed()* function:
  - ⭐ 0 Element is used for Throw Audio (Whoosh).
  - ⭐ Other Elements can be used for any Custom Game Logic (e.g., for Customizing Collisions with different surfaces (floor, wall, etc.). Example with using outside the Throwing System dynamically and uniformly: AR Throw & Score (docs)).

Colliders

Rotation (Torque) & Center of Mass

There are some nuances with Rigidbody.centerOfMass when rotation.

Symmetric Mesh

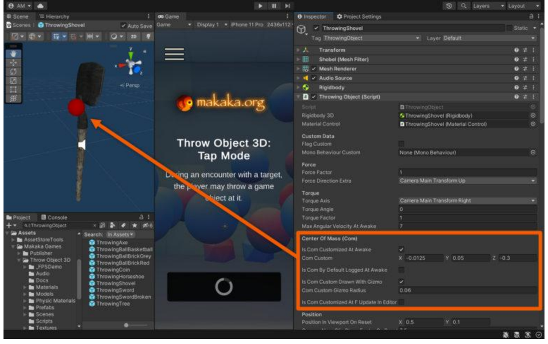
If you use Mesh Collider for Throwing Game Object and your mesh is ideally symmetric, then it's the perfect case and the game object will throw naturally.

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.

Solutions:

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 and Tools for Debugging it Visually (you can also turn on the Gizmos in the Game view to facilitate the debugging). There is no magic function that calculates the right Center of Mass.


makaka.org
Throw Object 3D:
Tap Mode
During an encounter with a target, the player may throw a game object at it.

ThrowingShovel
Tag ThrowingObject Layer Default
Transform
Shobel (Mesh Filter)
Mesh Renderer
Audio Source
Rigidbody
Throwing Object (Script)
Script ThrowingObject
Rigidbody 3D ThrowingShovel (Rigidbody)
Material Control ThrowingShovel (Material Control)
Custom Data
Flag Custom
Mono Behaviour Custom None (Mono Behaviour)
Force
Force Factor 1
Force Direction Extra Camera Main Transform Up
Torque
Torque Axis Camera Main Transform Right
Torque Angle 0
Torque Factor 1
Max Angular Velocity At Awake 7
Center Of Mass (Com)
Is Com Customized At Awake ✓
Com Custom X -0.0125 Y 0.05 Z -0.3
Is Com By Default Logged At Awake
Is Com Custom Drawn With Gizmo ✓
Com Custom Gizmo Radius 0.06
Is Com Customized At F Update In Editor
Position
Position In Viewport On Reset X 0.5 Y 0.1

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, you will have the right center of mass & right rotation.

Convex Mesh

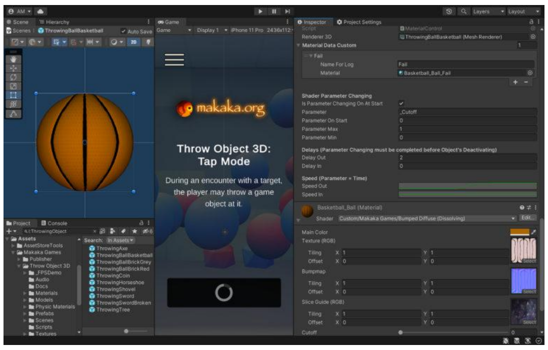
Unity allows you to use only convex mesh colliders. In some cases, it is not what you need because the collider by default can cover more space than the object itself in some points. In order to quickly & automatically get a convex mesh collider for complex objects, use Technie Collider Creator.

4. Material Control

MaterialControl.cs — an independent script that implements Fading (Dissolving) VFX for Throwing Objects: 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. Actually, you can change any Float Parameter in the shader over time.

Default Materials for Throwing Object is still placed in the Mesh Renderer component.


makaka.org
Throw Object 3D:
Tap Mode
During an encounter with a target, the player may throw a game object at it.

Script MaterialControl
Renderer 3D ThrowingBallBasketball (Mesh Renderer)
Material Data Custom 1
Fail
Name For Log Fail
Material Basketball_Ball_Fail
Shader Parameter Changing
Is Parameter Changing On At Start ✓
Parameter _Cutoff
Parameter On Start 0
Parameter Max 1
Parameter Min 0
Delays (Parameter Changing must be completed before Object's Deactivating)
Delay Out 2
Delay In 0
Speed (Parameter = Time)
Speed Out
Speed In
Basketball_Ball (Material)
Shader Custom/Makaka Games/Bumped Diffuse (Dissolving)
Main Color
Texture (RGB)
Tiling X 1 Y 1
Offset X 0 Y 0
Bumpmap
Tiling X 1 Y 1
Offset X 0 Y 0
Slice Guide (RGB)
Tiling X 1 Y 1
Offset X 0 Y 0
Cutoff 0

Parameters

- ⭐ *materialDataCustom* – array for Customizing Material Changing for Any Custom Game Logic (e.g., fail, win). Access from *ThrowControl.cs* or *ThrowingObject.cs* with *SetMaterial()* function.

  SetMaterial() function is used instead of Color Changing with *SetColor()* function to avoid memory allocations. So if you plan to change the Colors of Throwing Objects in the Runtime outside the Throwing System, you need to create Materials with Target Colors in advance.

  Example with using outside the Throwing System dynamically and uniformly: AR Throw & Score (docs).

Shaders

There are 3 Custom Shaders with "Slice Guide" Texture for Fading of Game Object and its Shadow.
Check *[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 Throw Object 3D 5.0+.*
> *Tutorial for the previous version can be found only in the asset folder.*

Getting Started with Throw Object 3D

Folders & Files 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 2021.2.13 & “3D” Template.
2. File > Build Settings > iOS, Android, Standalone (PC, Mac) or WebGL > Switch Platform.
3. Download and import Throw Object 3D into Unity.
    1. Warning Windows:
        1. Click “Import” to overwrite the Project Settings with predefined ones.
        2. Click “Install/Upgrade” for Package Manager Dependencies.
4. Next Packages are provided with Unity Package Manager, and they are already installed for this Asset by default. If packages are missing (Warning Window did not appear) then install them again with Unity Package Manager (with advanced settings enabled: “Pre-release Packages” & “Show Dependencies”):
    1. TextMesh Pro 3.0.6:
        1. **Always Required**: Window > TextMeshPro > Import TMP Essential Resources.
5. Reopen Unity Project.
6. Open Scene: Makaka Games > Throw Object 3D > Scenes > Demo.
7. Test in the Unity Editor or Build for Mobile or Desktop.

Creating your own Throwing Object Prefab

1. Create Prefab with a template:
    1. Drag one of the customized Throwing Object prefabs from the Project window into the Scene view,
    2. Name your Game Object,
    3. Drag the named Game Object into the Project window to create the Original Prefab.
2. Select the Game Object on the Scene & Customize it:
    1. Indicate Mesh of your 3D Model in Mesh Filter component,
    2. Customize Material in Mesh Renderer component (duplicate previous Material in the Project window and indicate here to separate different Throwing Objects),
    3. Customize Fading in Material Control component,
    4. Customize Colliders:
        1. Indicate Mesh in Mesh Collider component (or in more suitable collider component);
    5. *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 (*ThrowingObject*) for Prefab for convenient searching in the Project window.
7. Add Prefab to Random Object Pooler component of *ThrowingPoolControl* Game Object.
8. Throw Object Forward in "Play Mode".

Testing

Read Article: Mobile Testing.

Tested with Platforms

- ⭐ iOS on iPhone XS Max;
- ⭐ Android on Samsung Galaxy A71;
- ⭐ macOS;
- ⭐ Windows;
- ⭐ WebGL in Google Chrome.

Support

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

Changelog

Check the current version of Throw Object 3D on Asset Store.
The latest versions will be added as soon as possible.

5.0:

Features:

- ⭐ Dynamic and Unified Operating of Data for Different Throwing Objects (example with using outside the Throwing System: AR Throw & Score (docs)):
  - ⭐ *audioDataCustom* – Array for Customizing Collisions with different surfaces (e.g., floor, wall) or for Any Custom Game Logic; merged with deprecated *Whoosh Audio*. Access from *ThrowControl.cs* or *ThrowingObject.cs* with *PlayRandomSoundDependingOnSpeed()* function.
  - ⭐ *materialDataCustom* (*MaterialControl.cs*) – Array for Material Changing for Any Custom Game Logic (e.g., fail, win). Access from *ThrowControl.cs* or *ThrowingObject.cs* with *SetMaterial()* function.
- ⭐ *RandomObjectPooler.cs*: Split *areRandomizedObjects* boolean flag into 2 flags:
  - ⭐ *areRandomizedObjectsWhenCreating* – when you need a random instance count of possible Throwing Object prefabs every Scene Start.
  - ⭐ *areRandomizedObjectsWhenGetting* – when you require a random Throwing Object instance for every Throw.
- ⭐ Retrieving Count of Throwing Objects with *ThrowControl.cs* > *GetObjectCount()*.
- ⭐ Tools for Debugging the Center of Mass Visually in Unity Editor. If the Center of Mass by Default is not correct, you can use these tools for improving Center of Mass with Custom value.

Improvements:

- ⭐ Unity 2021.2.13.
- ⭐ Instantiating Throwing Objects at Start outside the Game Area in target *positionAtInit* (*RandomObjectPooler.cs*) — Avoid Collisions between Throwing Objects and with Other Physical Objects more accurately:
  - ⭐ Split Reset of Position:
    - ⭐ On Init – to Target Position at Instantiating.
    - ⭐ On Throw – to Camera Position.
  - ⭐ Split Layer Changing option into 2 stages & 3 states:
    - ⭐ On Init,
    - ⭐ On Throw & On Reset.
- ⭐ Consolidating the Same Code of Registering Throwing Object in Throwing System into *RandomObjectPooler.cs*.
- ⭐ *ThrowControl.cs* > *OnInitialized()* Event is raised more accurately: in the correct frame after all operations.
- ⭐ Shovel Throwing Object: Custom and Accurate Collider for Shovel.
- ⭐ Unified Menu Scene for All Scenes.
- ⭐ Loading Animation for Start Buttons.

4.4:

- ⭐ Unity 2021.1.20;
- ⭐ New Flag: "Is Throwing Blocked When Clicking UI";
- ⭐ Improve Project Settings to modern standards of New Unity Project;
- ⭐ Fix Resetting Rotation & Position from RigidBody to Transform to be compatible with AR Foundation camera (for AR Throwing (docs));
- ⭐ Fix Secondary Throw in "Flick" Mode when clicking on current Throwing Object after throw and before appearing of new throwing object;
- ⭐ Fix Throwing in "Flick" Mode when clicking outside any object with Collider after correct flicking throw;
- ⭐ Fix Throwing in "Flick" Mode when !isFullPathForFlick: clicking on empty space doesn't cause throwing now;
- ⭐ Fix "NullReferenceException" when "None" Game Object is assigned in Inspector of Random Object Pooler;
- ⭐ Fix Missing Textures.

4.3:

- ⭐ New Platform: WebGL Support (Tested in Google Chrome);
- ⭐ Unity 2021.1.12;
- ⭐ Fix Dissolving VFX.

4.2:

- ⭐ Optimization: TextMesh PRO for all texts.

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:

- ⭐ *AudioData* class and *PlayRandomSoundDependingOnSpeed()* function. Now you can easily declare Sound Settings in your own scripts to Play Sounds based on the speed of Throwing Objects. It's 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).