Asset Manual

Flappy Happy Unity Asset Overview and How to get started

Arcade Version - Overview

This package for Unity contains two parts. One is the Arcade machine, and the game that it runs. The other is a standalone version you can modify and use for mobile, pc or any other platform.

How to place the arcade game in the game
The arcade machine version can be placed as a prefab into your game scene.
You can find this prefab when you navigate to**:**
"Assets/Lowscope/Game - Flappy Happy/Arcade Cabinet Sample"

All the game contents can be found within the prefab. Expanding the prefab, you can see that it the game Is placed within the inactive "**Arcade Game"** object.

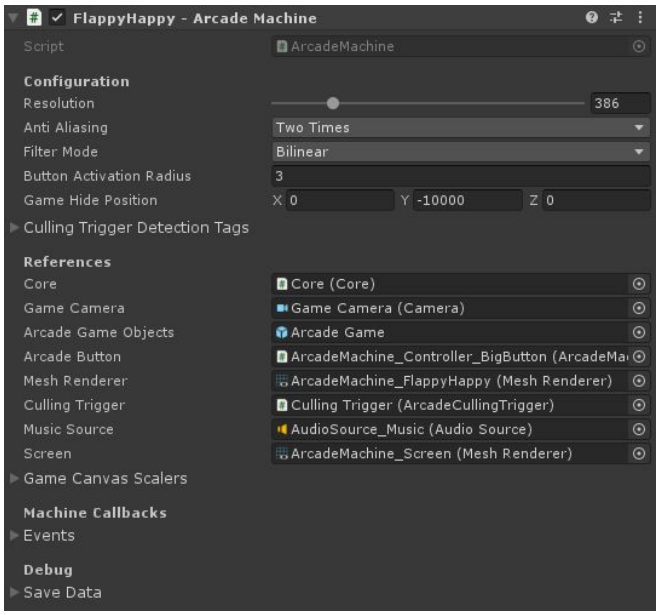
How does the arcade version work?
The arcade version uses all the objects the standalone Uses. With the exclusion of several UI elements and A game input component. The camera is being used to Send data to a render texture that is displayed On the arcade machine.

Tip when modifying the arcade version
In order to modify the arcade version extensively, it is advised to unpack the prefab and work on it in the testing scene. And then save it as a new prefab later on. For small changes you can just work on the prefab in the scene or directly.

How good is the performance of the arcade?
Several measures have been applied to improve Performance if the user isn't in the vicinity or looking at the arcade. Meaning, when no camera is looking at it, the game content gets disabled and no new render textures get rendered. Logic also stops. There is also a culling collision object. This uses a box collider to decide if it should render the game or not. For mobile you may have to set the arcade to a much lower resolution for it to be performant. In order to do this you can refer to the next page.

Arcade Version - Components

When selecting the prefab, you can see the main component that ensures the arcade game gets loaded. This component responsible for the base configuration. I will go through some of the configurations here to explain what they do.



1. **Resolution**
   The resolution of the render texture. Lower is more performant, but results in a more pixelated view.
2. **Anti Aliasing**
   Amount of anti aliasing that is applied to the render texture. Removes any jitter.
3. **Filter Mode**
   Point mode will make it fully pixelated, other modes will try to smooth out the texture.
4. **Button Activation Radius**
   Decides how far away the player must be before they are able to interact with the button. This happens through an **OnMouseDown()** event.
5. **Game Hide Position**
   The game that is parented to the object, and disabled. Will be placed at this location. The Reason for taking this approach is because Unity lacks good methods to dynamically cull objects.
6. **Culling Trigger Detection Tags**
   The trigger box will decide to show the game or not based on these tags. By default **Main Camera** and **Player** have been added. Change this based on your setup.
7. **References**
   Keep these as is. Unless you want to actively change the game.
8. **Machine Callbacks - Events**
   I will talk about these on the next page.
9. **Debug - Save Data**
   You can use this to inspect the currently saved data. All content gets saved as JSON.

Arcade Version - Event Callbacks

The arcade version provides a lot of callbacks. This allows you to add some fun secrets or mini-goals into the game. There are a total of 6 events you can choose from. Each event will send a UnityEvent with an integer as value. This for instance allows you to add a counter (setting text), or other things

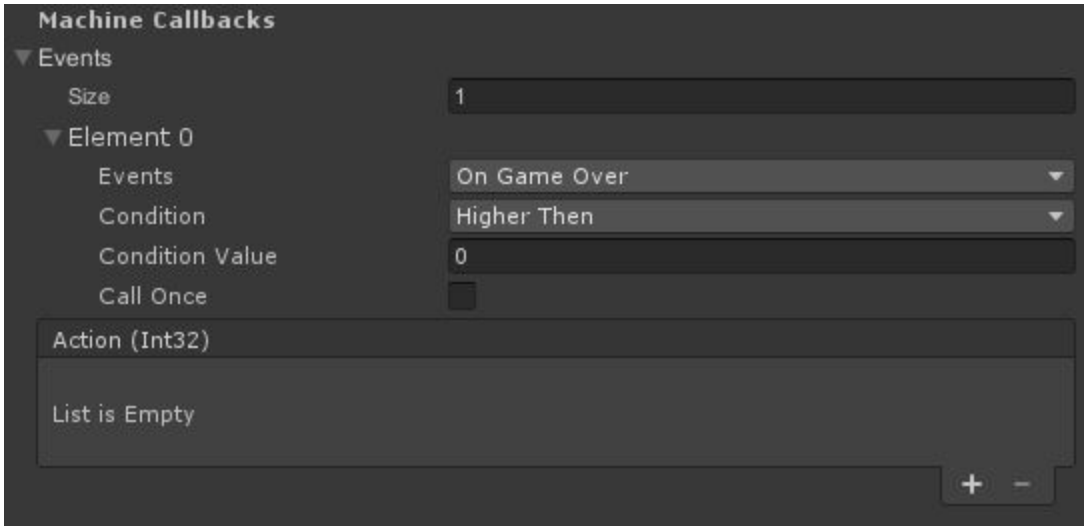
Events

1. **On Game Over** - Sends event based on times you have died.
2. **On Score** - Sends event based on times you have made a score
3. **On Play Duration** - Sends event after death, based on the seconds you played
4. **On Press Count** - Sends event based on total amount of presses
5. **On Start Game** - Sends event based on how many times you started the game
6. **On Total Points** - Sends event based on the total accumulation of points

Conditions may be very familiar for you when you code things. You do conditional checks based on the "Condition Value" to limit when an event is called.

Conditions

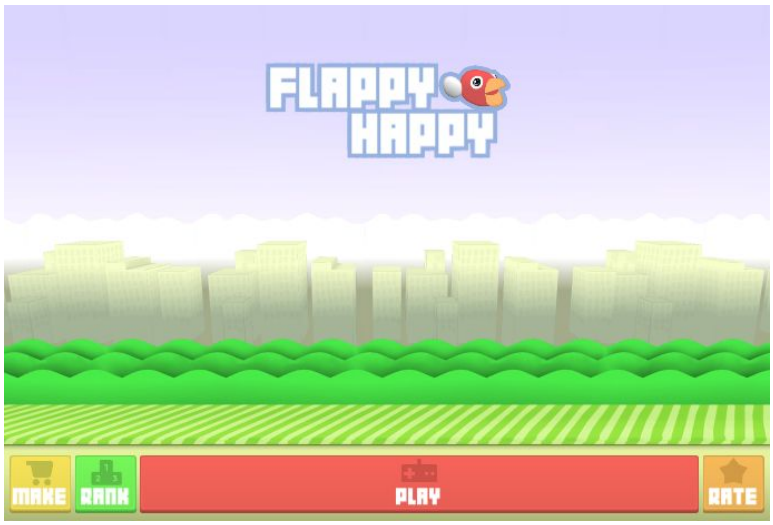
1. **Higher Then** - If **Event Count** is higher then **Conditional Value**
2. **Lower Then** - If **Event Count** is lower then **Conditional Value**
3. **Equal** - If **Event Count** is equal to **Conditional Value**
4. **Higher Equals** - If **Event Count** is higher or equal to **Conditional Value**
5. **Lower Equals**- If **Event Count** is lower or equal to **Conditional Value**

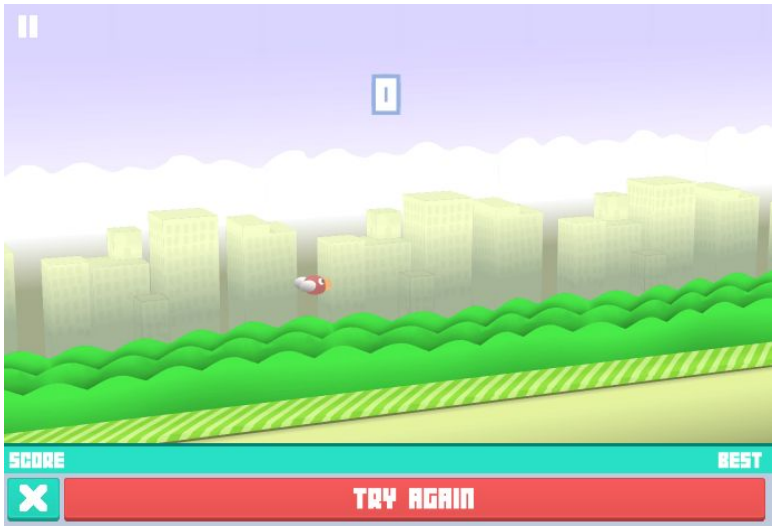


Regular Version - Overview

The regular version consists of two scenes. The main menu and a game scene.
The main menu is useful to redirect users to pages. Modifying these makes it easier to customize your game. Further. For instance the MAKE button could be replaced with a customization system.

Useful things to modify are





- **Buttons**
  The **GameButton.cs** Class contains all Logic that is required For the buttons. You can use this to call A menu or leaderboard.
- **Core**
  The main event distributor. Starting the game will call the core class and request a start game event.
- **Pipe Manager**
  This class is made to spawn The pipes. It automatically pools spawned pipes and calculates on what location to start spawning and despawning pipes.
- **Bird**
  You can find many parameters to modify on the bird component.
- **Shader Graph**
  If you go to the **"Game - Happy Flappy/Art/Shaders"** folder, you will see a range of shaders that were used to make the game a bit more unique. Like the moving bush and buildings & clouds.

Potential Questions and Answers

I want to add the arcade to a mobile game, does it perform well?

In order to test this, I have used a Samsung Galaxy A3 from 2016 as a benchmark. Using RenderTextures is expensive for mobile, lowering the rendertexture resolution allowed for better framerates (40+). You can also lower the resolution of the game textures and the arcade cabinet to improve performance even more. Another performance gain would be to use one material for all the objects in the flappy sample. However, it may give different results based on what game you place the arcade in. Therefore I cannot promise it will always run well. The settings that worked well for me were a resolution of 256, no anti aliasing and point filtering. The standalone version does work very well on mobile (60+ fps) . You can find a demo of this on the asset store page.

Is it hard to integrate the arcade machine into my game?

You can drag the prefab into your map and it should work. Although there are some things you may want to take into consideration. For instance, the arcade machine makes use of a trigger volume to save performance. Keep in mind that you have to set the correct tags in the arcade machine component on the root of the prefab.

Another consideration is input. Right now you can operate the arcade machine with OnMouseDown events. Meaning that you have to click on the machine. (There is a configuration for you to set the distance in the machine component). So if you have a FPS game with guns. You may want to implement your own interaction system.
In order to interact with the machine, you can call the PressButton() method. And you can toggle the “Use On Mouse Down” field on the arcade machine component in the inspector.

I want to use multiple machines, is this possible?

Yes. The only thing you have to take into consideration in this case is to set the “Game Hide Position” to a different position. To ensure the content does not collide.

What rendering pipeline does this asset use?

The arcade supports both Standard and URP. The standalone version was made with a focus for URP/LWRP. This is because it uses Shader Graphs for some of the shaders.
It still works for Standard, but it lacks some vertex offset effects.

Contact

If you have any questions regarding this product, you can send an email to:
info@low-scope.com