Color Switch Clone

"Complete Project Template"

I would like to start by saying thank you for downloading this Complete Project Template. Please rate it if you get the time… its mean a lot to me. I would also like to give you a little background on HBB and myself... I still consider myself very new to Unity, and C#. I have no classical programming training, though I do get some scripting/batching sorts of jobs competed at work on a Technical helpdesk without issue. I am quite fond of Unity. I had always enjoyed working on little game mods when I was younger, and the idea of game making. Unity Technologies has kind of made that dream a reality for me. You may read some places that when it comes to indie game development… "Fake it, till you make it". That is an idea I have had to subscribe to whole-heartedly as a solo developer, with no experience, and no classical training. I can do all of the jobs necessary to bring an idea to life, but because I have to do all of the "jobs" the end product suffers. If I would have been able to spend all of my time devoting myself to a single discipline like modeling, texturing, animation, game-design, or scripting I would be much better off. Since It can be hard to find individuals as devoted to / or passionate about a specific project or idea I continue to slowly continue learning/tinkering.

My main goals for releasing these kit are twofold. 1.) I wanted a pipeline to deposit some of my partially completed projects, while still trying to find some people to work with. It helps me to continue to learn, and practice. 2.) I am always learning/reading about Unity… I remember when I was starting out about a year ago, that it was INCREDIBLY useful for me to take apart other people's work. I would scour the Asset Store and I had just about every free complete project I could find. I think that in addition to the UT projects that there could definitely be more FREE Complete Project Templates. So I plan on releasing

several of the projects I have laying around on my computer as free kits. Beyond that I will probably sell some too, but they will be super cheap.

If you have gotten this far into my ramble then I thank you. Most of the scripts in the project have pretty thorough/lengthy comments. I wish they were a little more concise, but… c'est la vie.

1.) Setup

Setting Up Gravity, Tags, Scene Order

- **Set Gravity to -15 m/s² on the Y Axis.** See Figure1

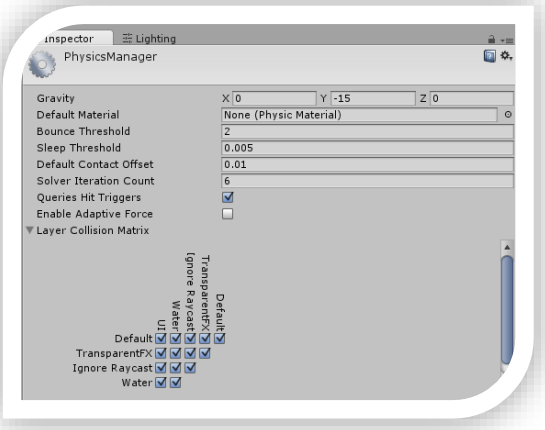


Figure 1 - No other changes are necessary in the Physics Manager.

- **There are several Tags used.** See Figure2

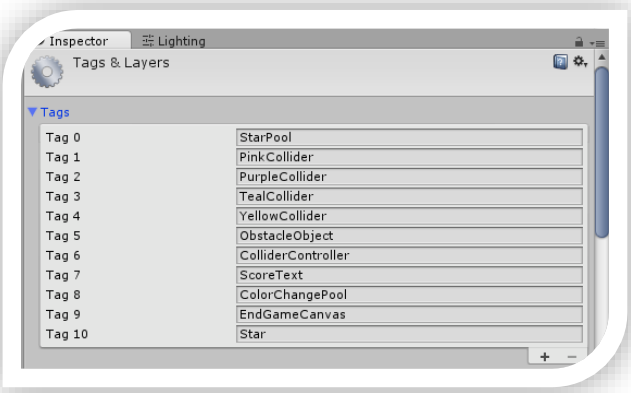


Figure 2 – These Tags are required.

The main FBX file that contains the majority of the primitive shapes I made, contain the obstacles. All of the obstacles are comprised of several pieces. This is

the individual pieces that will be colored, and colliders made for them. These Pink, Purple, Teal, and Yellow meshes make up our obstacles. Because some of them are more "rounded" shapes I decided to use the MeshCollider, for accuracy. When you add an obstacle to the scene, you color it (by adding a material if it does not have one), and then you add the color tag to it. The child meshes need to be individually tagged the color that you want to use. There is probably a better way to do this, though I could not come up with a quick solution. I grouped and selected the matching colored colliders and Batch tagged all of them, and then continued on to the other 3 colored colliders and repeating the process.

The Obstacle's Parent GameObjects need to be tagged "ObstacleObject".

The ColorColliderController needs the "ColliderController" tag.

The rest of the tags are pretty self-explanatory… The tags that end in "Pool" need to be appropriately assigned to the "StarPickup", and "ColorChange" pool gameobjects in the scene.

The objects in the starPool are tagged "Star"

The "EndGameCanvas" tag goes on the EndGameCanvas.

The "ScoreText" tag goes on the UI Text Object child of the GameCanvas.

- **Make sure Scenes are added to the Build Settings.** See Figure3

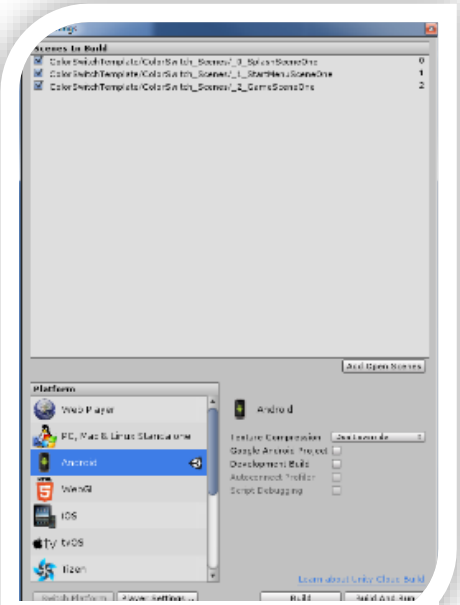


Figure 3 - Add all scenes to the build settings.

2.) Things to Know

A Basic Overview of the Obstacle Work

All of the obstacles that are used/re-used are in the scene at scene start. They need to be tagged "Obstacle Object". At runtime all of the gameobjects tagged "ObstacleObject" are collected, added to a list, and deactivated. Then 2 obstacles are enabled so that the players view (and slightly above), are occupied by obstacles.

The "ChangeColor" pickups spawn new Obstacles. So when the player collects the first "ChangeColor" item; obstacle 3 spawns. After the next one is collected; obstacle 4 spawns, etc.… The ColorChange pickup calls a method on the ColorColliderController (`IncrementObstacleProgression()`) that does this. The biggest part is designing obstacles that are passable. Take the Double Square obstacles as an example… If the outer square rotates 90 degrees on the Z Axis per second and the inner circle rotates -90 degrees on the Z Axis per sec, then there will be 2 occasions in which the player could traverse the path IF he/she is 1 of 2 colors. If the player was 1 of the 2 wrong colors when he/she came to that obstacle, then the fix would be to rotate the whole obstacle object 90 degrees to the Right or Left(if the shape is the right type of symmetrical shape). In this fashion we basically keep track of the fact that the obstacles spawn 2 ahead of the player. We check what color the player "will" be when they get there, by knowing what color they are NOW.

The `IncrementObstacleProgression()` method is where most of the magic happens. If you look inside the `IncrementObstacleProgression()` method to the switch statement with 22 cases (the number of obstacles I decided to make). In the switch you will notice that at Case(s) 3, 7, and 16 we pass a "`shouldWeRotateObstacle`" Boolean to the "`GetObstacle()`" method. That Boolean determines earlier in the `IncrementObstacleProgression()` method that YES, based on the players color we will need to rotate the whole obstacle by 90 degrees.

Now obviously not all shapes/combinations will work for this game, or with the "rotate obstacle 90 degrees" fix. The original game has a lot more variation in the obstacle types, but I just wanted to show a basic implementation. I did toy

with creating a spline to move colored dots along a path. The original does have an obstacle of that type, but I did not complete it. I knew this was going to be a free asset, so I did not sink a lot of time into the obstacle work. I will be updating this template here and there, and I may add them back in once I have time to finish/integrate the system.

3.) Change Player/Obstacle Color

Changing the Player and Obstacles Color

In the "ColorSwitch_Materials" folder you will find multiple materials, but four of the materials match the default obstacle colors. They are labeled Pink, Purple, Teal, and Yellow. The simplest most straightforward way of changing the colors is as follows. Select "Pink" and change the Main Color property to a new color. Now there are a few ways you can go about this next process, but I recommend this one…

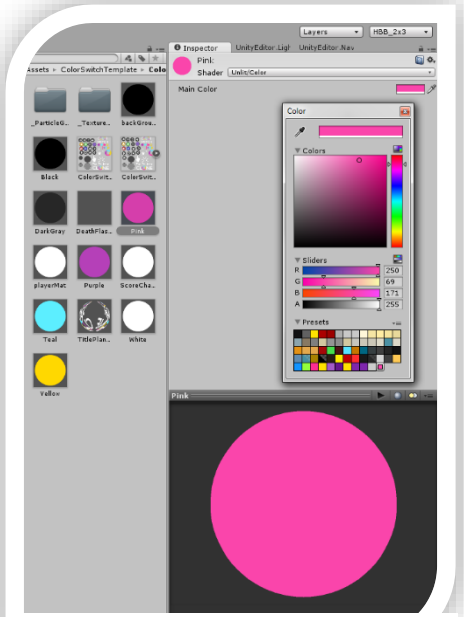


Figure 4- Change Obstacle Color/Save color as preset.

After the color picker is pulled up and you have the "Pink" material's color adjusted to the color you would like it to be, add it as a preset color towards bottom of the picker window (See Figure 4). Once it is added you can grab that sampled/saved color from the color picker at any time to use it. That is going to be important for us because later we will select the "Player", and change the colors in the "Player Color Array". Now do the same thing for Purple, Teal, and Yellow, remembering each time to add your new color as a preset. When you are done the last four presets will be your four new obstacle/player colors.

Now it is time to change the player's colors. The player GameObject is in Scene 2 (The main game scene). Select the Player and look at the "Player Controller" script in the inspector. Look for the "Player Color Array". Note that the colors may not be immediately visible if the "Player Color Array" is not expanded. Look for the small arrow and click it to expand the Player Color Array. You may notice that the colors are in the same order… Pink, Purple, Teal, Yellow.

Since we know that when the player is “Pink” he can pass “Pink” colliders we need to be sure to change the colors to the right color. ***I.e.*** whatever color “Pink” ***WAS*** changed to, needs to be the color that we change “Pink” ***TO*** on the player. The actual color of the player could be a little different, because the “matching colors” is just a visual representation of the 4 possible “*States”* the player is in… So to summarize, just make sure that if you changed “Pink” to green, that you change the first slot on the “Player Color Array” that WAS “Pink” TO green. Do this for all of the colors. Leave the last entry “Original Color” alone. You can (See Figure 5) … See that we changed the first 2 obstacle colors AND the first 2 Player Colors, to get an idea of what you should be doing. I know that may have sounded complicated, or I may have written it in a confusing manner, but just consider Pink, Purple, Teal, and Yellow as 0,1,2,3. When the player is 0, all of the obstacles that have the “Pink” Material assigned to them, have their Mesh Colliders disabled. When the player is 1, all of the obstacles that have the “Purple” Material assigned to them, have their Mesh Colliders disabled. Etc., Etc.… so just make sure color changes line-up. Otherwise, you can make the colors whatever you would like. I released a game with the colors changed, just to show that it was close to release ready (even If it lacked some exciting obstacles). I changed all the colors in the game, made the background and fader white, and then I named it Quad Color Jump. I will be using it to setup the Google Achievements and Leaderboards, and then I’ll update this package with the code.

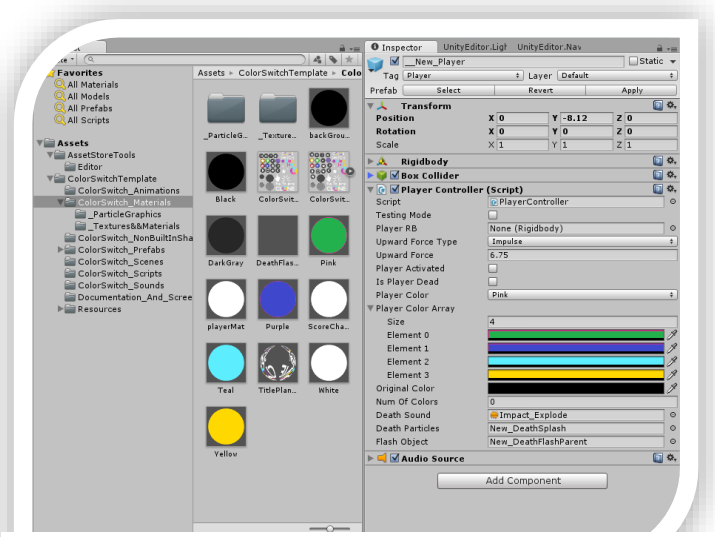


Figure 5 - First two obstacle & player colors changed

4.) Scripts

List of Scripts

- Most of the Scripts have fairly lengthy comments.

Please see the individual scripts for specific comments as to what they do. Everything should be fairly self-explanatory. All scripts are commented.

As I had said initially… this is intended to be a project that beginners can break down and learn from. One of the reasons I say this is because it is still in need of work. While it is intended for learning, you can use it in any way you see fit, no attribution required. I would however recommend that if you are considering using it for a release title that you customize/expand it. Whether you are a designer, or a developer, look into the code base and come up with ways to improve, expand, and customize it. At the moment it is not quite feature complete. I am releasing a version on the google play store which I am going to integrate Play Game Services into, but mostly so that I can then update this project with Play Game Services Integration. I initially struggled with Game Services integration (without using paid assets), so I think that having virtual currency, unity ads, and then achievements/leaderboards setup on the Play Services side would be a nice addition.

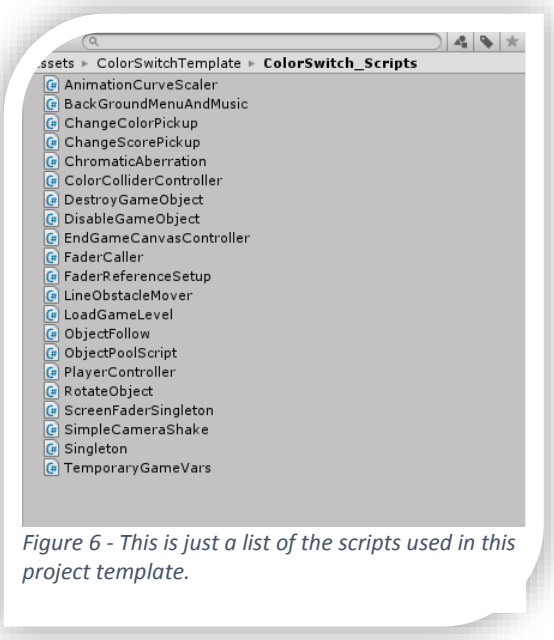


Figure 6 - This is just a list of the scripts used in this project template.

5.) Resolution and Scaling Canvas

Info regarding setup for specific devices and Issues users have had

When I initially released this kit I didn't really finish setting up all of the canvases/UI. It was my thought that since someone might have a desire to do something different with part of the template, or they would rather test the UI resolution with their devices resolution, that I would just let individual users play with the Menu/UI scaling and positioning.

Normally when I near the end of a development cycle for an app that is going on the marketplace, that is when I start doing advanced UI work and checking multiple devices (even if just 3 physical devices, and then about 10 emulators).

Then an application will go into alpha/beta channels and I sit back and wait for specific device issues/complaints. Up until I get to that point I decide on general layout, and the number of Buttons/Text fields earlier in development… the "design" part of the UI. I hadn't really got into that mindset because this wasn't necessarily a "ready to release" game template, but rather a fairly complete "vertical slice"… so I skipped/forgot that step, and I shouldn't have.

I have had several people e-mail me regarding the UI being stretched, not lined up, and other things being scaled incorrectly. So I made a written response to one individual, and then I decided to make a YouTube video showing how to set-up the project so that it "works as expected (desired)". I will link the E-mail and the YouTube link below. Before that I will detail some of the things to consider when using this game template. I designed this template to run at 480x800, Locked to Portrait Mode. The initial game was obviously portrait only. When I am in my Standalone Platform within unity I have two screen profiles that I added manually. Android Tall, and Android Wide. I often stay in standalone mode until I start testing on device, which happens late in my development cycle too. I have a super a cheap Samsung Galaxy Ace Style, and I use it as a device that is close to

my “bottom end” of compatible devices. Most devices nowadays are at least that fast or faster. I also test on a Galaxy Proclaim from time to time, but I’m getting off track... The point is that I give my editor ample space, and set it specifically to 480x800. What has been happening is that users are importing the Color Switch Clone Game Template and they are in Standalone Platform, and their Game View is set to “Free Aspect”. The first thing they see is that in the Menu Scene (scene 1), that the Round play button is GIGANTIC, and It is all misaligned. This happens on several devices or screen sizes. This is the Project pulled into the Editor – Android Platform – Samsung Galaxy S3 [See Figure 7].

The solution is simple. Most of us know how to manipulate the New UI system. Lol... It’s not really the “new” UI anymore. All of the canvases have Canvas Scaler Components on them, and when I initially released the template I left them set to “Constant Pixel Size”. I thought the users would do what they wanted with the UI, or that they would want to set it for their specific device. That led many people thinking there were errors, or it was broken, and truthfully it looked bad. So I set all the canvases in the game to “Scale by Screen Size”, and to Match Height Completely (Match set to 1f). Set the Reference Resolution to 480x800. That is my device resolution. This way if the device has the same aspect ratio it just scales the resolution up, or if its lower it scales the resolution down. If the aspect ratio does not match my devices aspect ratio (what I am setting everything in the project to (480x800)) then it will change the aspect ratio keeping the “height” in mind. So this way it may not stay at the reference resolution, BUT since the game is portrait mode ONLY, it will keep the height and the aspect ratio locked. Only changing the resolution. That way from top to bottom everything is on screen. Unlike in Figure 7. So now the worst case scenario is that the width of the application may be a little too narrow or a little too wide, but the

Figure 7- The 480x800 project on a 720x1280 Screen

main upward lane SHOULD be visible. The Graphic below (Figure 8) shows a before and after of the new UI scaling. The changes will be included in the game

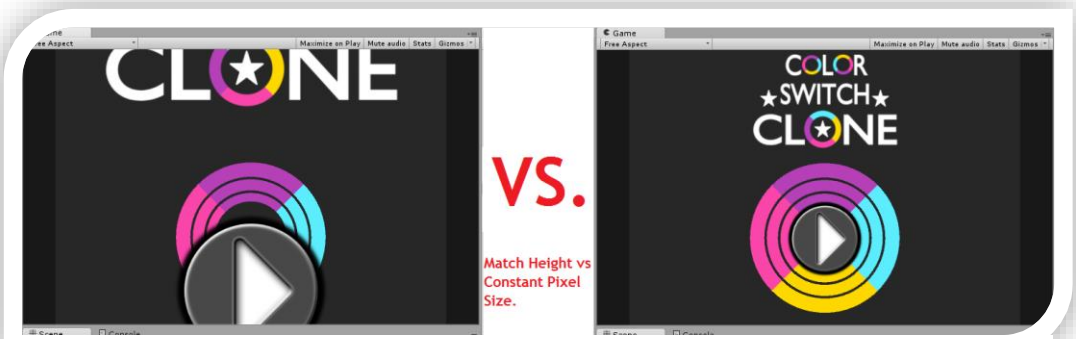


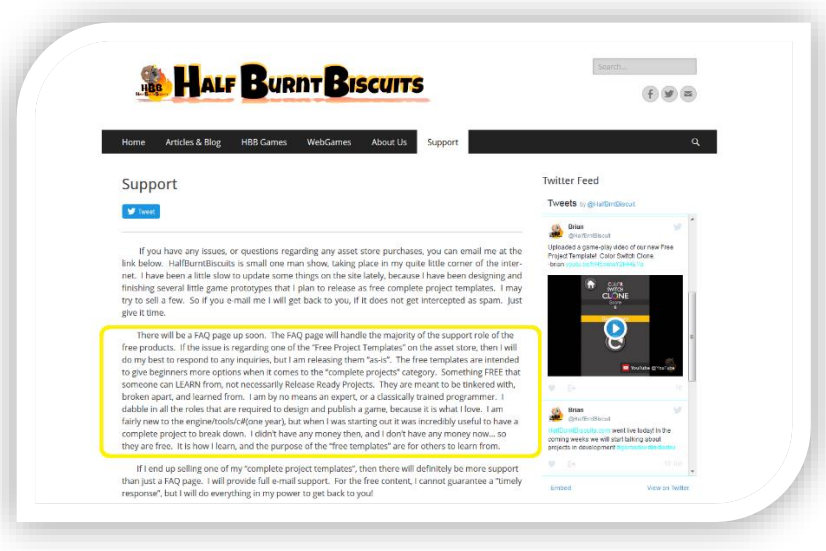
Figure 8 - Constant Pixel Size vs. Scale by Screen Size & Ref Res 480x800

template update. That way no one thinks something is messed up, or broken.

Everyone needs to adjust the UI to their liking, and test on various devices. More than likely the anchors of various UI Objects need to be set to different locations. Preferably some positioning would be managed via scripting, and anchors adjusted based on ScreenWidth, and ScreenHeight. Get the Center of the Screen by doing ScreenHeight / 2 and ScreenWidth / 2... Etc., etc. That way no matter the device you can you have the center of the display. Corners obviously work a little differently, but there are ways to do them almost perfectly as well. Other things can be deduced and anchored this way and paired with canvas scalers to create optimal resolutions and device independence. Research this topic. A good place to start is in Unity's UI How To section:

http://docs.unity3d.com/Manual/UIHowTos.html

If anyone has any comments, questions, or suggestions you can e-mail me at HalfBurntBiscuits@gmail.com. I will respond as soon as I can. I will eventually get a FAQ page setup, but as a solo developer it takes me time to take care of game design, making the game, and updating the website. The “free” Complete Project Templates are released “as-is”, and the majority of the “support” will be provided by the “FAQ”.



I will also be selling some of the Complete Project Templates I publish, but they will be super cheap, and e-mail support will be provided. I guess what I am saying is if you send me an e-mail, be patient. As long as the spam filter does not grab the e-mail, I will get back to you. It may however take a couple days.

Contact:

Brian

E-mail:

HalfBurntBiscuits@gmail.com: Comments and Suggestions are Welcome. This is the best way to contact me, with any issues.

Website: Working on getting some content uploaded. Site has a WebGL Demo of ColorSwitchClone.

http://www.halfburntbiscuits.com/

Link to YouTube Video showing how to change the Current UI issues, for people to use prior to the first update.

https://youtu.be/zigcHu-YU3o

Twitter: New Account. Once some more content is uploaded to the site, I will use this more.

@HalfBrntBiscuit

Docs Update with version 1.01.

Change-log

Changelog Version 1.01

- Fixed a typo in the Standalone Player Settings. The Max Screen Height was incorrect.
- Added a section in the Documentation regarding changing the color of the player, and obstacles per user request.
- Fixed an issue where a couple versions of Unity (between 5.0.1 and 5.3.3) after building to android the application would not recognize taps. The user could not hit the play button. The fix was to delete and re-add the EventSystem in each scene, save, and rebuild. Somehow the Touch Input Module was no longer on them, when it was required.
- Changed all canvases to “Scale by Screen Size” from “Constant Pixel”. I also changed the Reference Resolution. Many users were confused by the UI/Menu appearance when they

imported the project into the editor. A detailed written section and a link to a video is in the included documentation now. It discusses the current canvas setup and production settings.

- Removed leftover metadata that referenced my initial key-store alias. This caused the user to have to tick “Unsigned (debug)” or creating a key-store before initially building to android. It’s now back to default Unity state.