2D Action Platformer Kit Documentation

About this Starter Kit

This starter kit was designed to provide completely custom, and customizable examples for learning or getting a head start on building 2D Action Platformers. Read through the scripts, add on to them, and write new ones to build your own custom 2D Action Platformer.

All of the assets included in this file are free to use, but we recommend creating your own media to make your game unique. All of the scripts can be used as-is but we also recommend using them as a reference or build onto to create unique and great games.

1) Preparing Scenes for Building

To be able to compile the game you must add all of the scenes to the build settings. When adding scenes to the Build Settings, make sure the scene “loader” is at the top of the list in Build Settings.

> To do this, double click on the scene “loader”, then in Unity go to File>Build Settings...
>
> Then click “Add Current”. Once Loader is added, then do this same process with the rest of the scenes (menu, level1, level2, level3). Order doesn’t matter at this point as long as the scene “loader” is first. You can create as many levels as you want, but you have to add them to the build settings as well.

2) Building for Mobile

2d Action Platformer was designed to run in landscape. To to make sure that the game runs in landscape, go to File>Build Settings...

> Click on which mobile platform you want to build to (iOS or Android), then click Switch Platform.
>
> After this, click Player Settings... On the right, options for player settings will pop up.
>
> Click on the Android or iOS tab (depending on which one you chose) below “Per-Platform Settings”.
>
> Click on the drop-down menu to the right of Default Orientation* and choose Auto Rotation. Then options below will appear to choose which orientations to allow.
> Make sure only Landscape Right, and Landscape Left are chosen.

3) Building for Web/Standalone

You do not need to tamper with orientation like mobile for web/standalone builds. Make sure to follow the steps from “Preparing Scenes for Building”. Otherwise you are will be ready to build as many versions for web/standalone as you’d like!

4) Inputs and Controls

- Web and Desktop Controls -
The player’s controls to move left, right and jump reside in playercontrols.js. The default controls for Web and Desktop builds are both WASD, and Left Right. Jump is Spacebar. These can be changed in the playercontrols.js script. Notice there are 2 iterations of these controls for #if UNITY_WEB and UNITY_STANDALONE. What this does is it will only compile those pieces of the script, depending on what platform you have your build set to.

The player fires bullets through the script weaponsystem.js. The default for firing bullets is Left-Shift for both Web and Desktop builds. These can easily be changed in playerweapons.js. Notice there are also 2 iterations of shooting controls in #if UNITY_WEB and UNITY_STANDALONE. The reasons are the same as described above.

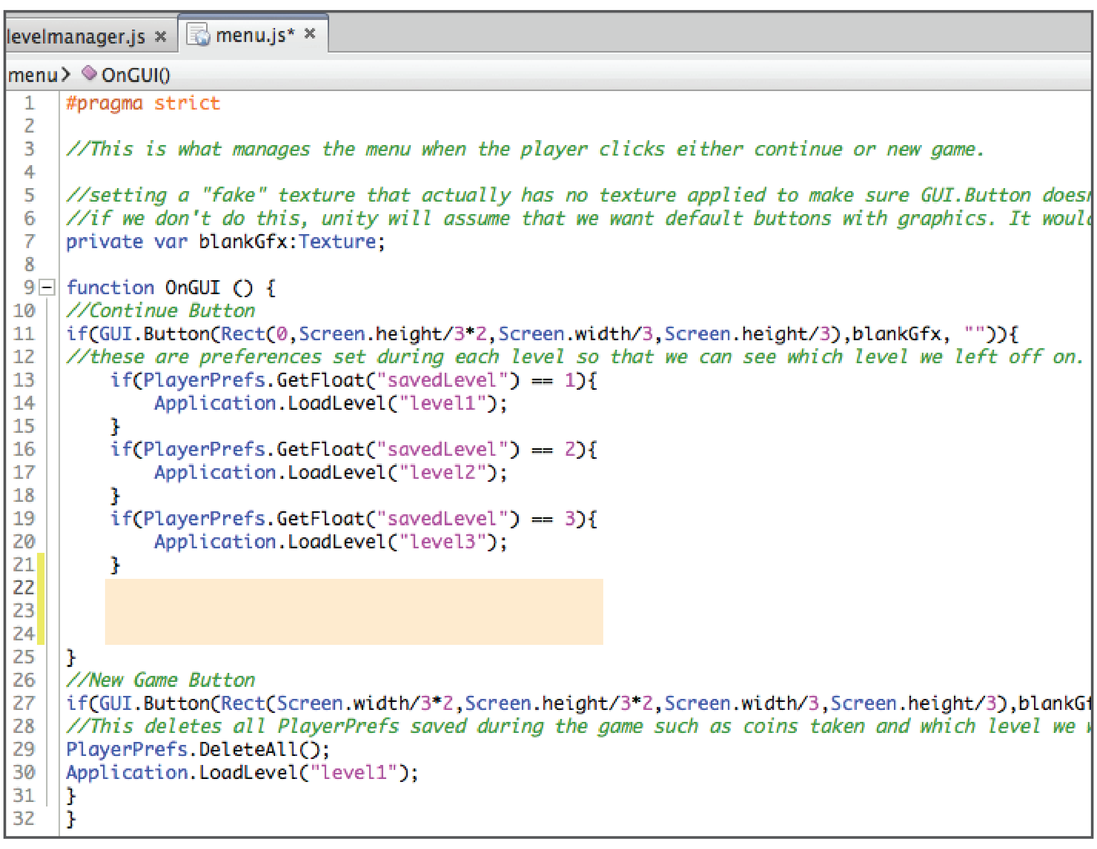
- Mobile Controls -
Mobile Controls are also in playercontrols.js with 2 iterations within #if UNITY_ANDROID and UNITY_IOS. These controls can be edited in any way you’d like, but default works out of the box.

The player on mobile versions fires bullets through the script playerweapons.js. To change how the bullets fire for mobile, they can be changed there in both iterations of #if UNITY_IOS and UNITY_ANDROID.

5) Building More Levels

We tried to make it as easy as possible to build onto the kit. If you are trying to add more levels to the default game, here is a simple list that will allow you build as many as you’d like.

- Go to a pre-existing scene that is a level, and copy all of the contents in the hierarchy.
- Make a new scene by clicking File>New Scene.
- Delete the camera that is in that new scene, then paste the contents that you just copied.
- In the hierarchy, go to GUI>levelfont. Change it to the level you’re making. (Ex. Level 4)
- In the hierarchy, click on the “endlevel” object. Options should come up in the inspector. Here you need to set the number of level you’re on under the script “levelmanager.js” for the variable Level Number. Then write in the name of the next scene you want to load into the variable Next Level. (Ex. level5, menu, credits, etc.)
- Save the scene now. Name it what level it is (Ex. level4).
- Then go to the level previous to this one (Ex. If you’re creating Level 4, go to Level 3) and edit the same variable Next Level, and set it to the name of the level that you’re currently creating (Ex. level4).
- Then go into the Scripts folder in the Assets folder 2dactionplatformer. Find the script menu.js. Double click on it so it opens in the script editor.



```
#pragma strict

//This is what manages the menu when the player clicks either continue or new game.

//setting a "fake" texture that actually has no texture applied to make sure GUI.Button doesi
//if we don't do this, unity will assume that we want default buttons with graphics. It woul
private var blankGfx:Texture;

function OnGUI () {
//Continue Button
if(GUI.Button(Rect(0,Screen.height/3*2,Screen.width/3,Screen.height/3),blankGfx, "")){
//these are preferences set during each level so that we can see which level we left off on.
    if(PlayerPrefs.GetFloat("savedLevel") == 1){
        Application.LoadLevel("level1");
    }
    if(PlayerPrefs.GetFloat("savedLevel") == 2){
        Application.LoadLevel("level2");
    }
    if(PlayerPrefs.GetFloat("savedLevel") == 3){
        Application.LoadLevel("level3");
    }




}
//New Game Button
if(GUI.Button(Rect(Screen.width/3*2,Screen.height/3*2,Screen.width/3,Screen.height/3),blankGf
//This deletes all PlayerPrefs saved during the game such as coins taken and which level we
PlayerPrefs.DeleteAll();
Application.LoadLevel("level1");
}
}
```

- Add this chunk of code to the area in menu.js thats highlighted in orange above. Then edit the 3 thats red below and the string “level3” thats highlighted in red below, to match the level number and level name you created. This will allow the level you just created to be saved as progress if the player stops playing during that level.

```
if(PlayerPrefs.GetFloat("savedLevel") == 4){
        Application.LoadLevel("level4");
}
```

And thats it. Your level will now be set up and can be played through normally in the game. If you’re having any issues with this please let us know.

Contact Us

If you have any issues, questions, or suggestions, please do not hesitate to contact us!
Email us at any time at support@cinoptstudios.com and we will get back to you as soon as we can.