Extra Unity Topics

By Peter Mitchell

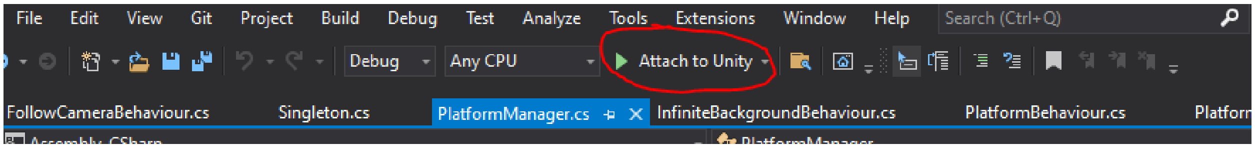
Contents

1 Using Debugger with Unity

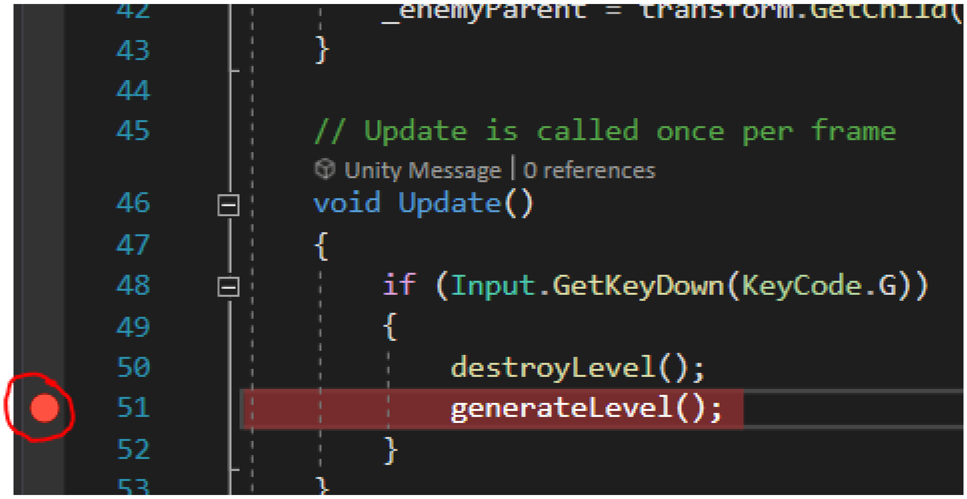
You can view additional information at:

https://docs.unity3d.com/Manual/ManagedCodeDebugging.html

1. Click Attach to Unity in Visual Studio.



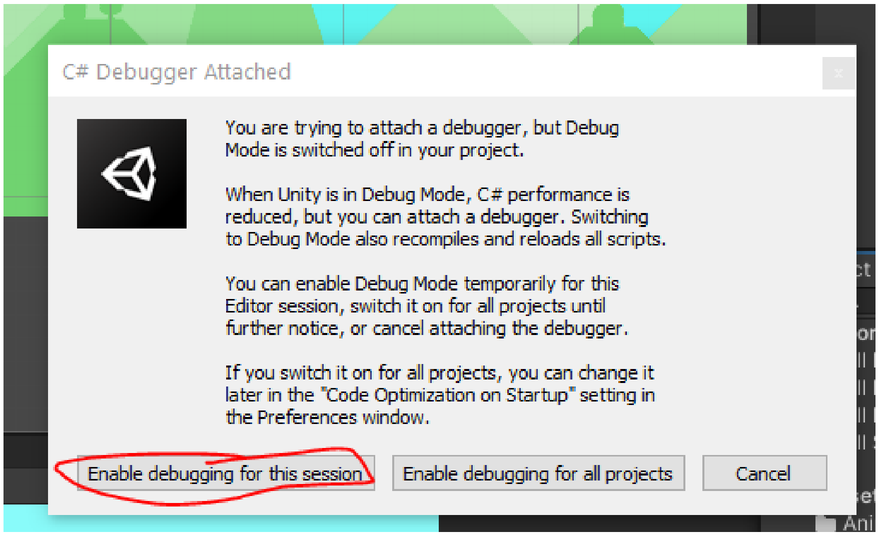
2. Create a Breakpoint.
   You can either click in that far left differently coloured column or you can select some code, Right click it, then Breakpoint->Insert Breakpoint.



```
            _enemyParent = transform.GetChild(
        }

        // Update is called once per frame
        Unity Message | 0 references
        void Update()
        {
            if (Input.GetKeyDown(KeyCode.G))
            {
                destroyLevel();
                generateLevel();
            }
        }
```

3. When you return to Unity you will find a popup. Click Enable debugging for this session.



C# Debugger Attached

You are trying to attach a debugger, but Debug Mode is switched off in your project.

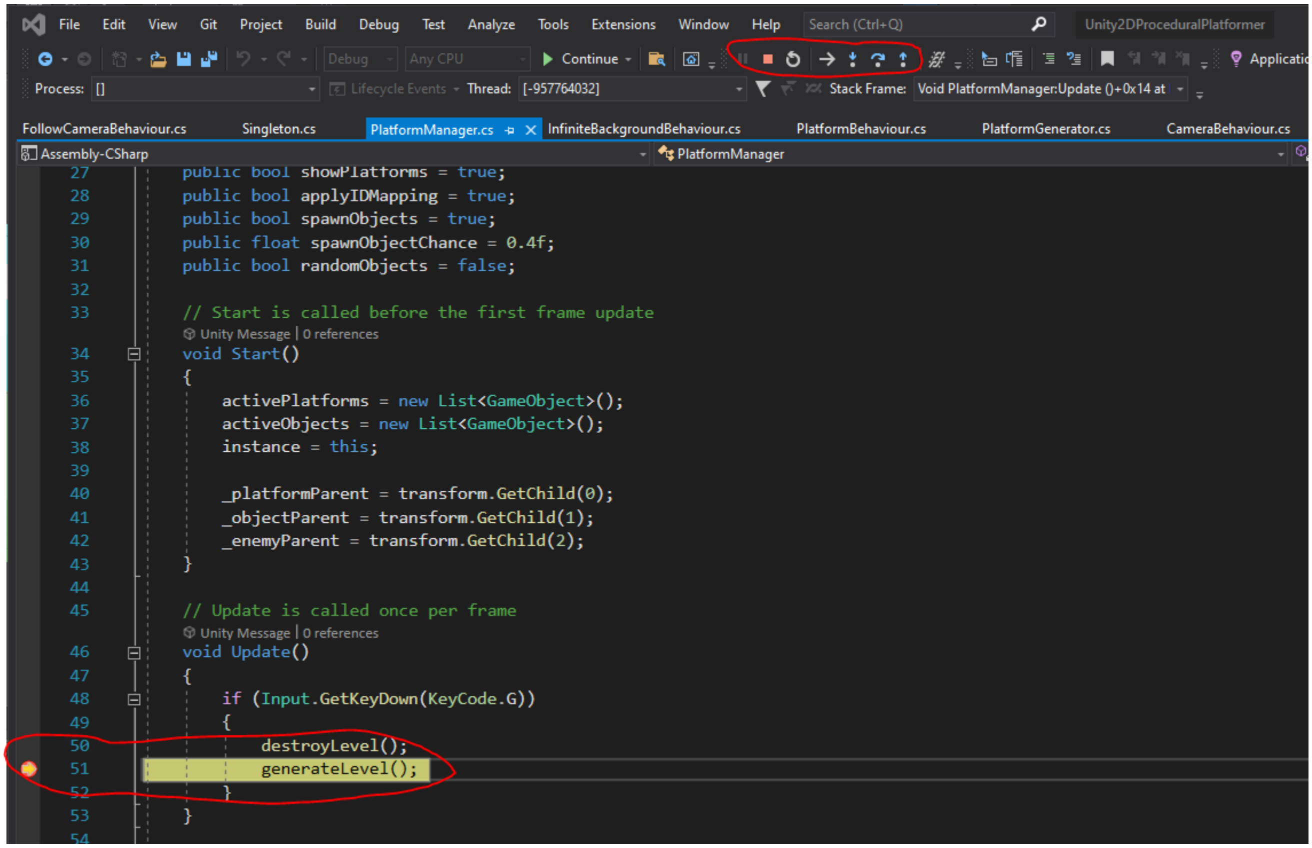
When Unity is in Debug Mode, C# performance is reduced, but you can attach a debugger. Switching to Debug Mode also recompiles and reloads all scripts.

You can enable Debug Mode temporarily for this Editor session, switch it on for all projects until further notice, or cancel attaching the debugger.

If you switch it on for all projects, you can change it later in the "Code Optimization on Startup" setting in the Preferences window.

Enable debugging for this session
Enable debugging for all projects
Cancel

4. Run your game. And do something that triggers the Breakpoint you set. Note that it will likely have to recompile your scripts to allow debugging.
5. You can see below that the program has paused at the Breakpoint. In the top is a circled set of controls. And you can see the line is highlighted to show where the program is paused currently.



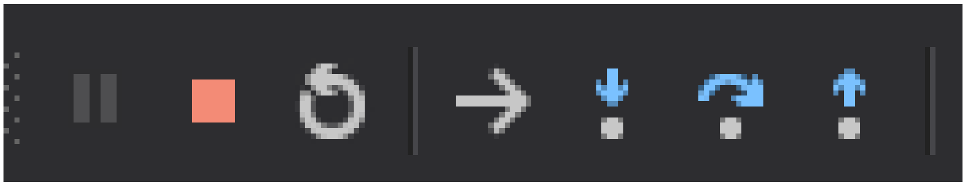
```
    public bool showPlatforms = true;
    public bool applyIDMapping = true;
    public bool spawnObjects = true;
    public float spawnObjectChance = 0.4f;
    public bool randomObjects = false;

    // Start is called before the first frame update
    void Start()
    {
        activePlatforms = new List<GameObject>();
        activeObjects = new List<GameObject>();
        instance = this;

        _platformParent = transform.GetChild(0);
        _objectParent = transform.GetChild(1);
        _enemyParent = transform.GetChild(2);
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.G))
        {
            destroyLevel();
            generateLevel();
        }
    }
```

6. The controls are shown larger here:



In order:

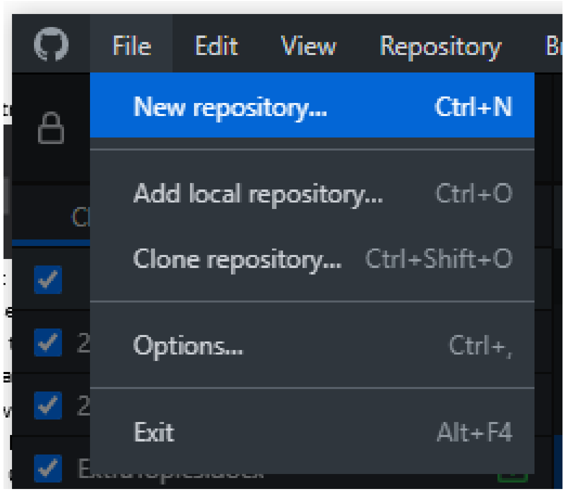
- Pause (to pause at a current point in execution).
- Stop to Terminate.
- Restart to Restart the Application.
- Show Next Statement.
- Step Into.
- Step Over.
- Step Out.

And if you just want to continue execution as normal you can click “Continue” over where the Attach button was. You will mostly probably use Step Into to move inside functions, and Step Over to move to the next step of a function.

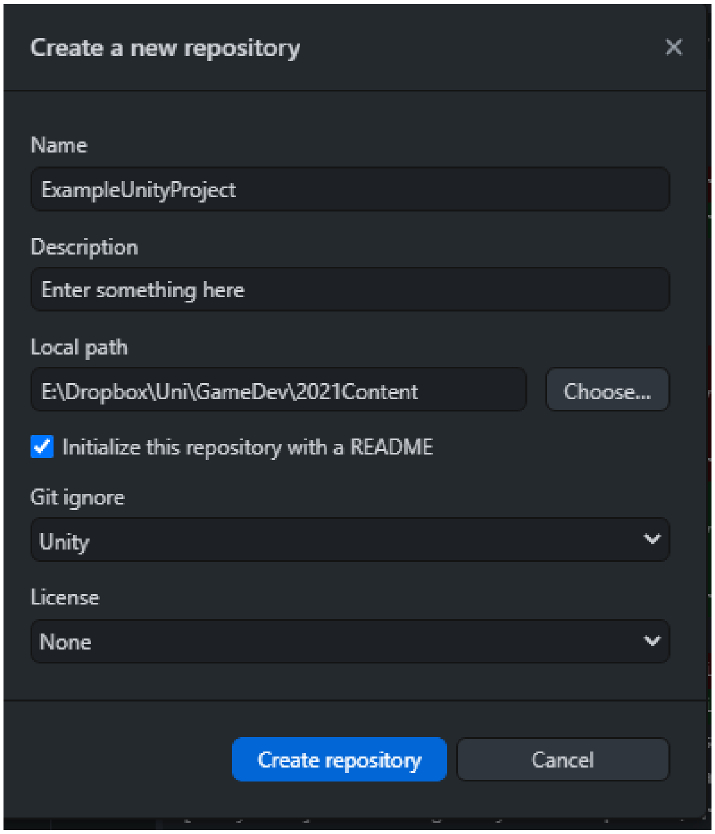
You will find down the bottom associated variable states and the Call Stack. In combination these allow you to track down the cause of bugs by watching your variables change as you move through the program line by line.

2 GitHub Unity Project Setup

1. Create a GitHub account and download the GitHub Desktop application. You can use the command line to control everything with GitHub if you wish instead, but if you are doing that you probably don’t need this guide and should be able to find the resources yourself.
2. Create a new Repository.



3. You will see a dialog as seen below.



Create a new repository

Name: ExampleUnityProject

Description: Enter something here

Local path: E:\Dropbox\Uni\GameDev\2021Content

☑ Initialize this repository with a README

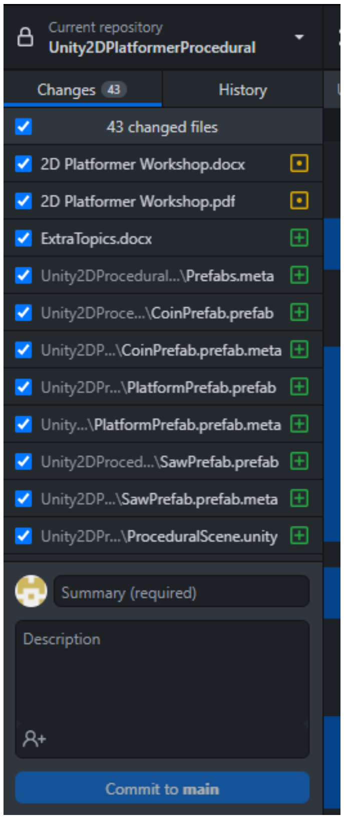
Git ignore: Unity

License: None

Make sure to set a project name, description, a path where you want the project to be placed on your system, (optionally) tick the Initialise with README, and VERY importantly select "Unity" from the Git ignore dropdown. This will exclude files that are generated by Unity and are not necessary to the project being used on multiple systems.

Click Create repository when ready.

4. Either create your Unity project inside the folder you chose (I would suggest a nested folder). Or copy your Unity project folder into it.
5. You will see a list of all the added/changed files as shown to the right.
6. Write a summary that describes the content you are adding and if necessary, a description to further elaborate. Then click the Commit to main button.
7. When you are ready to make the changes appear on GitHub you can use either of the "Push origin" buttons as seen over page.



Current repository
Unity2DPlatformerProcedural

☑ 43 changed files

☑ 2D Platformer Workshop.docx

☑ 2D Platformer Workshop.pdf

☑ ExtraTopics.docx

☑ Unity2DProcedural...\Prefabs.meta

☑ Unity2DProce...\CoinPrefab.prefab

☑ Unity2DP...\CoinPrefab.prefab.meta

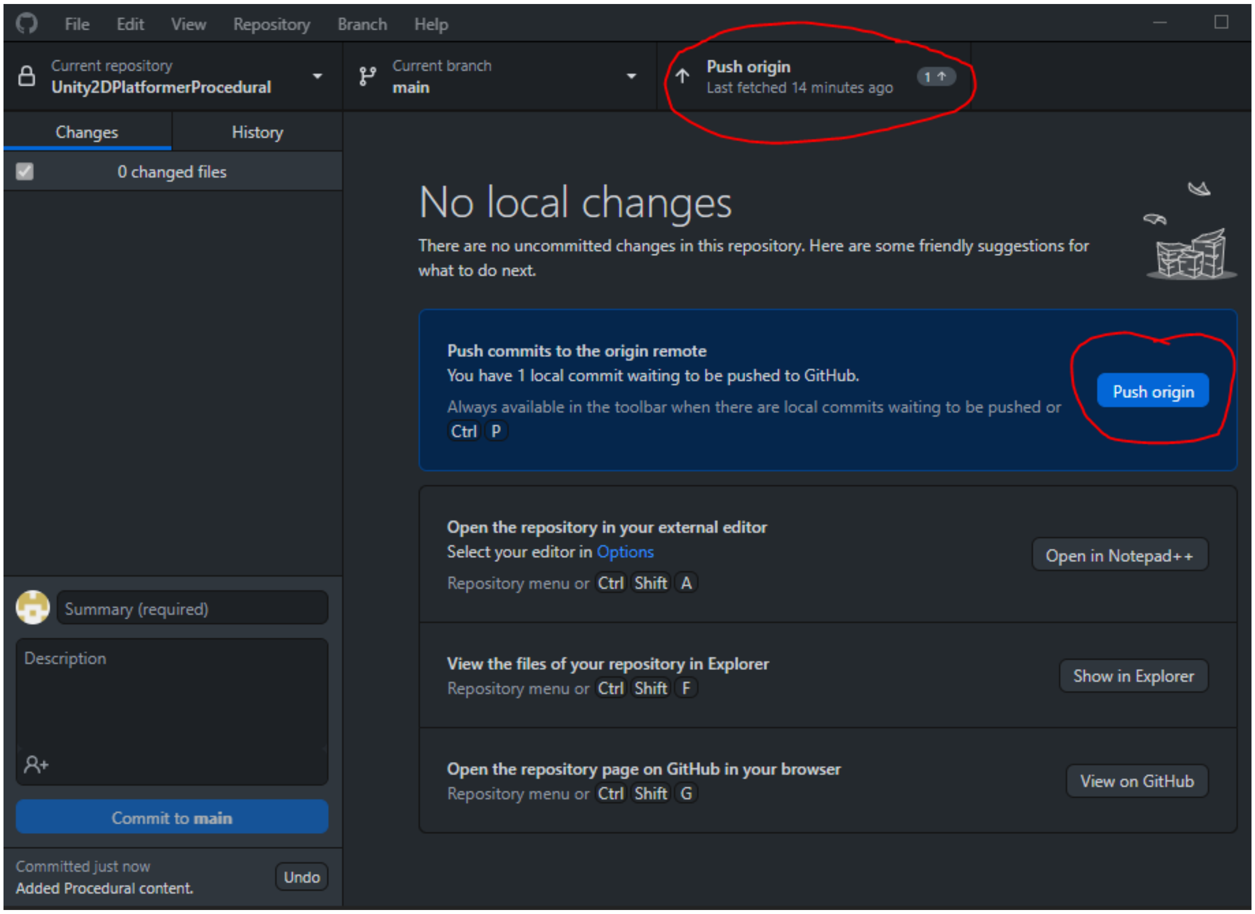
☑ Unity2DPr...\PlatformPrefab.prefab

☑ Unity...\PlatformPrefab.prefab.meta

☑ Unity2DProced...\SawPrefab.prefab

☑ Unity2DP...\SawPrefab.prefab.meta

☑ Unity2DPr...\ProceduralScene.unity



Current repository Unity2DPlatformerProcedural

Current branch main

Push origin
Last fetched 14 minutes ago

0 changed files

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Push commits to the origin remote
You have 1 local commit waiting to be pushed to GitHub.
Always available in the toolbar when there are local commits waiting to be pushed or Ctrl P

Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl Shift A

View the files of your repository in Explorer
Repository menu or Ctrl Shift F

Open the repository page on GitHub in your browser
Repository menu or Ctrl Shift G

Committed just now
Added Procedural content.

2.1 Note about GitHub and Large Files

If you are dealing with files near 100MB or more, you will have to use Git LFS as part of your setup. You can find information about this below.

https://docs.github.com/en/github/managing-large-files/versioning-large-files/configuring-git-large-file-storage