Before we do anything, I want to give a quick shout out to my latest Patreon supporter, Coburn 64. Thank you very much man, I greatly appreciate the support.

(did you just assume my gender?)

I did conduct a poll on Discord about which video topic to discuss and everyone that voted wanted me to do a shooting mechanic in Unity, CryEngine and Unreal Engine. However, Due to circumstances of real life, I was not able to finish that script for this week, so we will be doing another topic today.

Hello! I’m Jesco, you’re watching game dev made easy and today, we are going to discuss how to install Python 3 in Visual Studio Community Edition 2017 on Windows and how to get started with PyGame.

(new intro)

Now before we get started with this, I want to preface that Python is not a language I typically use for a myriad of reasons but the reason why Python is a great choice of a language to use is due to the fact that rapid prototyping is possible with the language. When you add PyGame to the mix, which in its own rights has become a popular game development tool, you can prototype out different game mechanics and create a great proof of concept for any game development projects you are working on. This is not to say that you can’t prototype and build a full game in PyGame with Python as that is something you most certainly can do. For those of us that use C# with Unity or CryEngine or C++ with CryEngine or Unreal Engine, we know the pains of trying to prototype and build a proof of concept with these engines and Python with PyGame can most assuredly ease those pains significantly.

PyGame is a 2D game library that is truly portable. You can build games for just about everything that you can think of as long as there is support for it. It was built on top of the SDL library, it has multi core cpus in mind, has optimized c and assembly code for core functions, works with multiple graphics backends, and is very easy to use.

With the wall of text finally taken care of, let’s actually get started with installing Python and PyGame.

(transition)

The first thing you want to do is open the Visual Studio Installer Application. And if you’re like me and need to update the Visual Studio Installer, go ahead and take care of those updates.

(an eternity later)

Alright, we are back and now we can continue on with this. With your installation of Visual Studio, click on modify. With workloads, scroll down a little and locate Python development. For installing everything, it will take up about 4 and a half gigs of space but the only things you absolutely need are Cookiecutter template support, Python web support and Python 3 64 bit or Python 3 32 bit. Once those have been installed, we can continue on.

Open up Visual Studio Community Edition 2017. Click on File, New and Project. If you navigate to the Python folder in the Project Window, you will notice many different templates to choose from and only one of them offers Pygame but that is with PyKinect. So, instead click on Python Application as the type and click ok.  
  
Click on View and scroll down to Other Windows to show more window options. Locate and click on Python Environments. Just under Python 3.6 64 bit, you should see Overview, click on it and switch it to Packages PyPi. Just underneath, the Packages Pypi or pie pie, there is a text box, click on that and type PyGame inside of it. In the output window, it will show you that has downloaded PyGame and installed it.

Switch over from Python Environments to the Solution Explorer. Inside you will see Python Environments added to the Python Application in the Solution. Click on Python 3.6 and it will reveal pip, pygame and setup tools. We now have Python and PyGame installed. Now we can have some fun and create a little demo application to make sure everything works appropriately.

(transition)

It is extremely important to remember that Python is whitespace and tab sensitive. This is part of the reason I don’t like to use Python often as it goes against what I am used to with C#, C++, C, F# and Assembly. To preface this, when I was making this simple example we are about to go over, I made the mistake of not correctly tabbing and caused unnecessary havoc to debug and fix.

The very first thing to do is to import pygame. This is to actually gain access to the pygame library and to be able to call functions made available to us. Next up is to write pygame dot init. This initializes the pygame modules that will allow it to run appropriately.

We need to now define the width and height of our window. To do this write dWidth and set the value to be 800. This will define the width of the window we will create to be 800. Write dHeight and set the value to be 600. This will define the height of the window to be 600. Now we have the width and height as 800 by 600 resolution.

We can now use these values to set the actual display window size to be this resolution. To do this, type gameDisplay is set to pygame dot display dot set underscore mode with the parameters of dWidth and dHeight.

With every application and game, it normally has text with it that says the name of that program. We can do this with pygame as well. Type pygame dot display dot set underscore caption with the parameter being a string value of Game Dev Made Easy Tutorial. This will set the window’s name to be whatever the string value is.

Now, we need to define the image that we want to be displayed on screen. Since I am a bit egotistical. (cough) I mean, I have set the convention to be my logo, we will continue with that. Write gdmeLogo and it will be set to pygame dot image dot load, it will have the value of Jesco dot png.

Now we need to define the logo itself as it will be an image displayed on the screen. Write def logo with the parameters of x and y and put a colon after it. This should automatically indent the next line, if it doesn’t, press the tab key one time which should put the cursor to be lined up with the “l” in logo. Write gameDisplay dot blit with the parameters of gdmeLogo comma with x and y in another set of parenthesis. Blit allows you to draw one image into another and it is considered a surface in pygame. A Surface is a pygame object for representing images.

This next part is extremely important. Make sure that you are no longer indented with the next set of items we write. If you are indented, then Python will think that everything that is indented is part of the definition for logo and you will have the application throw errors every time you go to test the program.

We will define what the x and y values will be next. X should be set to 150. Y should be set to 50. This will put the image to be roughly in the center of the screen.

We now want to create a Boolean value with the name of ended and it should be set to false. We will be using this as a means of exiting a while loop.

And speaking of that while loop, let’s go ahead and create it.To do this, write while not ended with a colon. This is our game loop.

You should again be indented. We need to create a for loop at this point. This for loop will be as followed: for event in pygame dot event dot get. This checks to see which events have been called and how to handle certain events. Write if event dot type is equal to pygame dot quit with a colon. Ended is now set to true. This will exit the while loop upon ended being changed to true.

Make sure your indent is lined up with the for loop for this next line. Write logo with the parameters of x comma y. This will draw the logo onto the screen.

And with any game, we need an update loop so write pygame dot display dot update. This will update the display every frame.

For the next part, remove all indenting.

Write pygame dot quit. This calls the quit sequence to exit and deinitialize the pygame module.

Write quit. This will exit the python application environment itself.

The very last thing is to add the image into visual studio. So add either my logo or any other image by right clicking on the Python Application in the Solution Explorer, highlight add and then click on Existing item. Make sure to change from just python files to all files to locate the image you want to use. Click okay after you’ve selected the image you want and the image will be imported into the solution.

Press the play button which should have the word attach next to it in visual studio. This will call the python runtime environment and then it will launch pygame.

As we can see, we have an 800 by 600 resolution window with the phrase, “Game Dev Made Easy Tutorial” as the window’s title and has the Game Dev Made Easy logo in the center of the screen. You can also see that the total amount of code utilized was very minimal so rapid prototyping is very possible using pygame and python.

As stated before, using Python and Pygame is very good for testing and prototyping. Even if you mainly use a 3D only engine, this can be useful. For example, if you were working on testing out new algorithms for creating something, conversion from 2D to 3D is very simple and you can use this sort of environment to test how they would run and perform. This has been Jesco with Game Dev Made Easy and remember, use every tool you can and always strive to learn something new. You never know when it may become useful for you.

(new outro)

You’re still here? There’s nothing else here for you to see. Well since you are still here, I have multiple social media websites that you could follow me on such as Twitter, Quora, Minds or even Gab. Well, go on! Shoo! click on one of the videos on the screen for a choice of what you could watch next and you are most welcome to bug me on social media or discord.