بسم الله الرحمن الرحیم

به نام خداوند جان آفرین ، حکیم سخن در زبان آفرین

## مقدمه

Godot ، یک نرم افزار بازی سازی یا اصطلاحا Game Engine رایگان و منبع باز است که توسط دو برنامه نویس به نام های Ariel Manzur و Juan Linietsky و به وسیله زبان برنامه نویسی قدرتمند C++ توسعه داده شده است . سازندگان Godot آن را تحت گواهینامه ی رسمی MIT ( MIT License ) در اختیار عموم قرار گرفته است .

گواهینامه ی MIT یکی از آزادترین گواهینامه های نرم افزاری است ، بنابراین تحت این گواهینامه شما اجازه دارید :

- ✓ برنامه ها ( بازی هایی ) که توسط Godot می سازید را بدون پرداخت هیچ گونه حق استفاده ای به سازندگان Godot ، به هر صورتی ( رایگان ، تجاری یا ... ) به اشتراک بگذارید
- ✓ شما می توانید کدهای منبع Godot را تغییر بدهید ، به آن ویژگی های جدید اضافه کنید و یا حتی با استفاده از آن یک Game Engine جدید با نامی دیگر بسازید و منتشر کنید .

این Game Engine برای سیستم عامل های مختلف از جمله Windows ، Mac ، Linux عرضه شده و با استفاده از آن می توانید بازی هایی برای پلتفرم های مختلفی مثل Android ، Windows ، iOS ، HTML 5 بسازید .

سازندگان Godot ، آن را اینگونه توصیف کرده اند :

« Godot یک Game Engine پیشرفته با ویژگی های بسیار زیاد است که می توان از آن به عنوان پلتفرمی برای توسعه ی بازی های دو بعدی و سه بعدی استفاده کرد . این Game Engine مجموعه ی بزرگی از ابزارهای پایه ای و ضروری برای توسعه ی بازی ها در اختیار شما قرار می دهد که می توانید به وسیله ی آن روی طراحی بازی تمرکز کنید نه اینکه مجبور باشید چرخ را دوباره اختراع کنید ! »

البته Godot بیشتر برای ساخت بازی های دوبعدی و بازی های دو نیم بعدی و نهایتا بازی های سه بعدی بسیار ساده کاربُرد دارد و نباید انتظار داشته باشید که این Game Engine همان امکانات و قدرت Game Engine های قدرتمند سه بعدی رایج را در اختیار شما بگذارد .

به خاطر ساختار نسبتا خاص این Game Engine ( در ادامه با آن آشنا می شوید ) یادگیری و کار با آن ساده است و با کمی وقت گذاشتن می توانید بر آن مسلط شوید و تقریبا هر بازی دو بعدی که خواستید را بسازید . البته باید توجه داشته باشید که در هنگام نوشتن این آموزش فرض کردم که شما حداقل با یک زبان برنامه نویسی رایج ( سی ، جاوا ، سی شارپ ، پی اچ پی ، پایتون یا ... ) آشنا هستید و مفاهیم پایه ی برنامه نویسی رویه ای و شی گرا را می دانید و با مفاهیمی از جمله ی Event System آشنا هستند . در طی آموزش سعی کرده ام همه چیز را به صورت شرح دهم و تا جای ممکن کمی در مورد این مباحث صحبت کنم ، ولی اگر کاملا تازه کار هستید ، بهتر است در مورد اصول برنامه نویسی کمی تحقیق کنید و بعد به سراغ بازی سازیی بیاید ، زیرا بازی سازی به خاطر طبیعت ذاتا پیچیده و پویایی که دارد ، جزء سخت ترین مباحث در برنامه نویسی است .

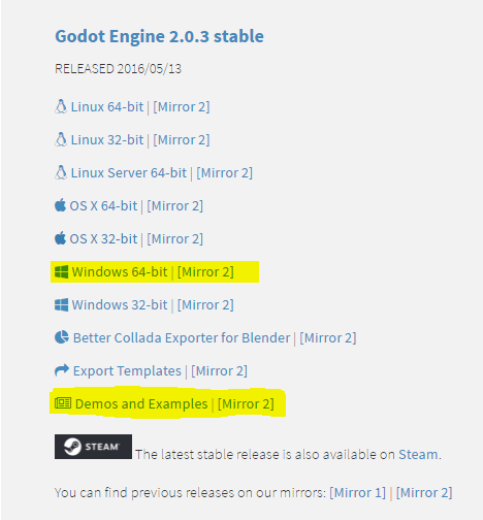
## دانلود ، نصب ، و ساخت اولین پروژه در Godot

برای دانلود Godot به بخش دانلود وبسایت رسمی Godot به آدرس زیر مراجعه کنید :

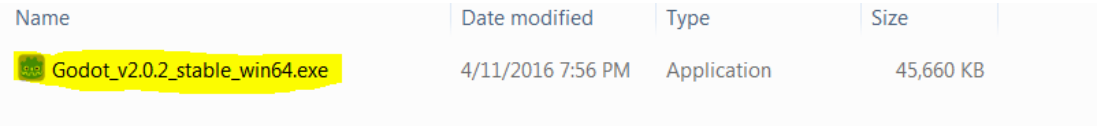
http://www.godotengine.org/download

من فرض می کنم که شما از سیستم عامل ویندوز 64 بیتی استفاده می کنید ( رایج ترین سیستم عامل نصب شده روی رایانه های کاربران ایرانی ) ، برای همین با کلیک بر روی دو لینک زیر Godot و دموهای آن را دانلود کنید :

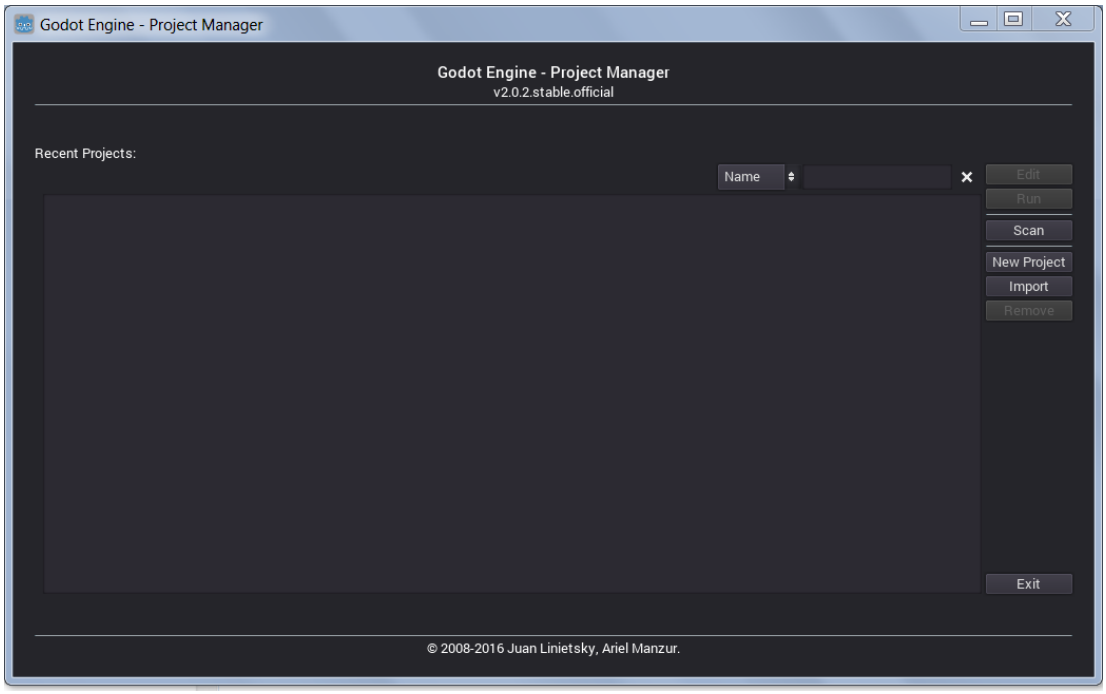
Windows 64-bit _ Demos and Examples




پس از دریافت فایل های فوق ، آنها را در دو پوشه ی جداگانه در مسیر دلخواهتان ( درایو و پوشه ی مورد نظرتان ) از حالت فشرده خارج کنید . ما از نسخه ی 2.0.2 این برنامه در هنگام نوشتن این آموزش ، استفاده می کردیم .

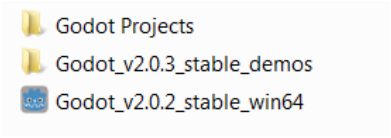



شاید انتظار داشته باشید که فایل نصبی Godot را ببینید ولی نیازی به نصب Godot نیست ! فقط کافی است بر روی Godot.exe دابل کلیک کنید تا Project Manager برنامه باز شود .

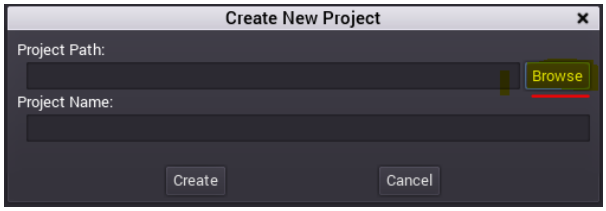



این اولین باری است که Project Manager را باز می کنیم ، بنابراین لیست پروژه های آن خالی است . می خواهیم اولین پروژه را بسازیم و جمله ی کلیشه ای Hello World را چاپ کنیم ! ولی قبل از آن یک توصیه می کنم ، به محل قرار گیری Godot.exe بروید و یک پوشه جدید برای

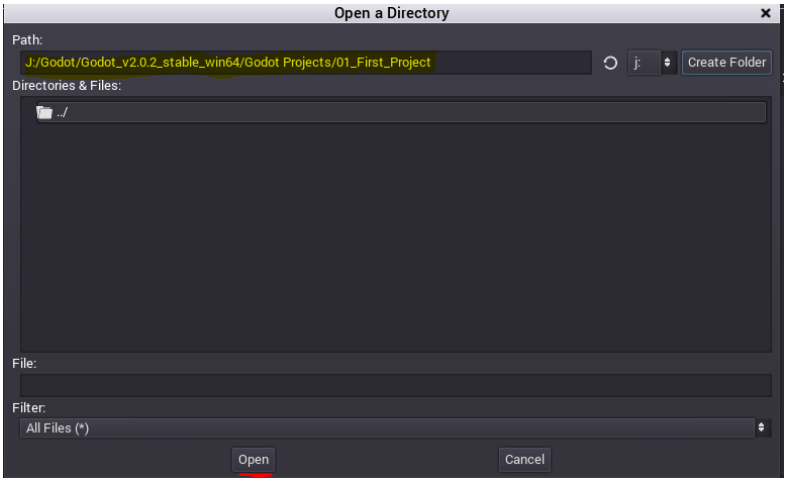
پروژه های Godot بسازید . ( من یک پوشه ی جدید به نام Godot Projects ساختم و پوشه ی پروژه های تمرینی Godot را به این مکان منتقل کردم. )



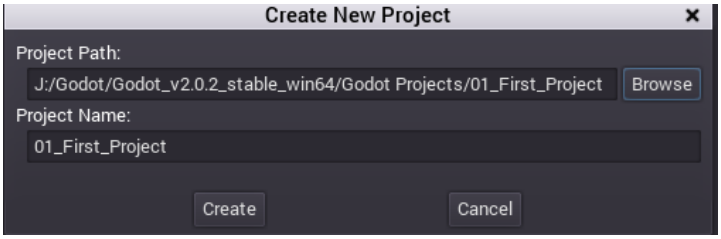

خب ، برای ساخت پروژه ی جدید بر روی دکمه ی New Project کلیک کنید تا پنجره ی Create New Project باز شود :



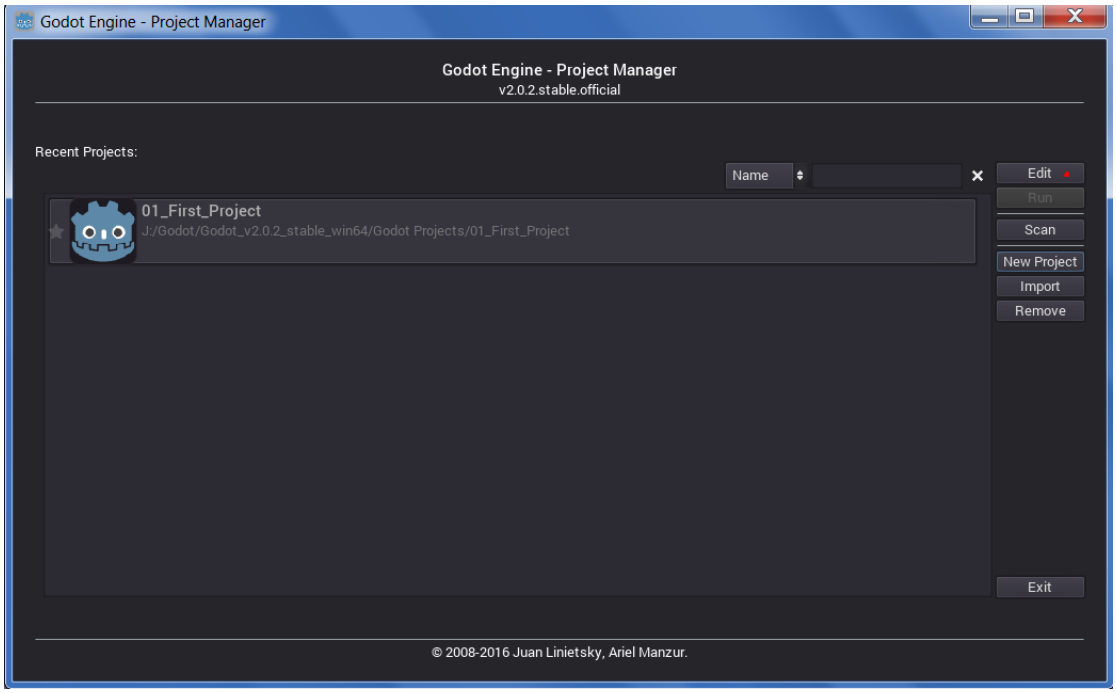

با کلیک بر روی Browse ، Project Path ( محل قرار گیری پروژه ) را انتخاب کنید . من از طریق با رفتن به پوشه ی Godot Projects و کلیک کردن روی دکمه ی Create Folder یک پوشه به نام 01_First_Project ساختم و با وارد شدن به این پوشه ، آن را به عنوان محل ذخیره ی پروژه انتخاب کردم .



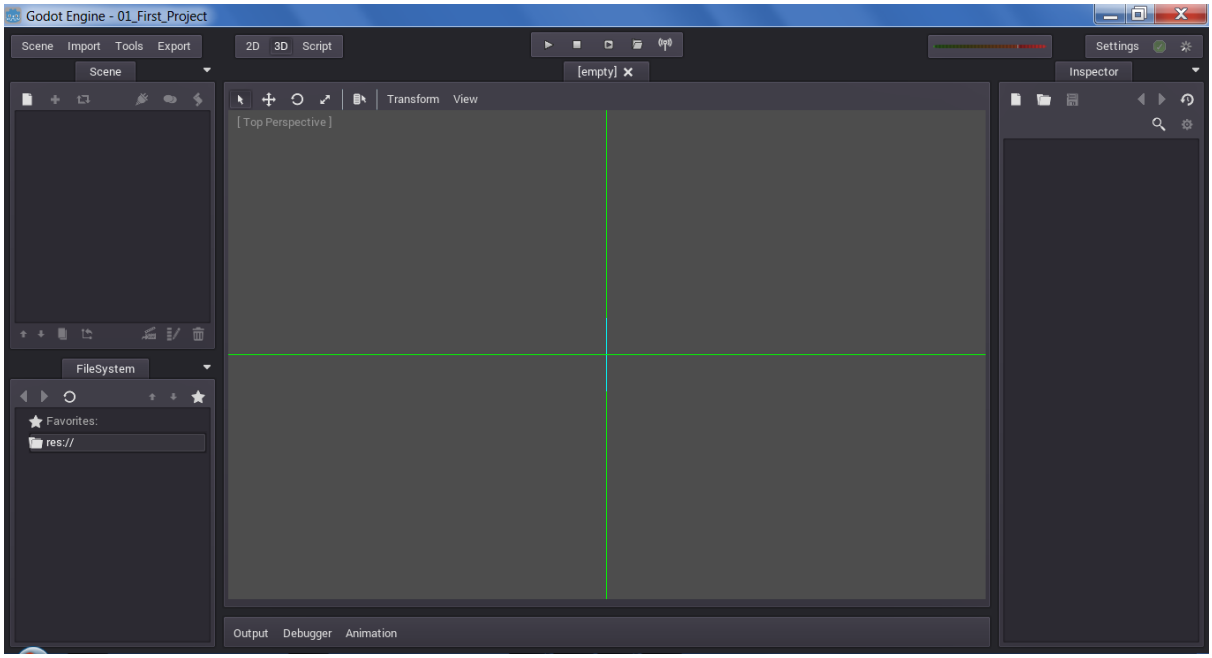

حالا در پنجره ی Create New Project فیلد های Project Path و Project Name به صورت خودکار تکمیل شده اند ، می توانیم یک نام دیگر برای پروژه در نظر بگیریم ولی من از همین نام پروژه راضی هستم ، پس بر روی دکمه ی Create کلیک می کنم.، تا پروژه ساخته و به لیست پروژه های Godot در Project Manger اضافه شود .



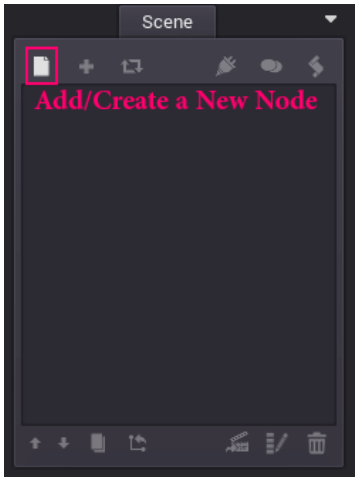
حالا می توانیم اعمال مختلفی از جمله ، حذف کردن یا اضافه کردن یا وارد کردن پروژه های جدید را در Project Manager انجام بدهیم . برای ورود به پروژه ای که ساختیم و ویرایش آن ، پروژه را انتخاب کنید و روی دکمه ی Edit برنید ( یا روی اسم پروژه ، دابل کلیک کنید . )




پس از اینکار وارد محیط ویرایش پروژه یا اصطلاحا Editor می شویم .

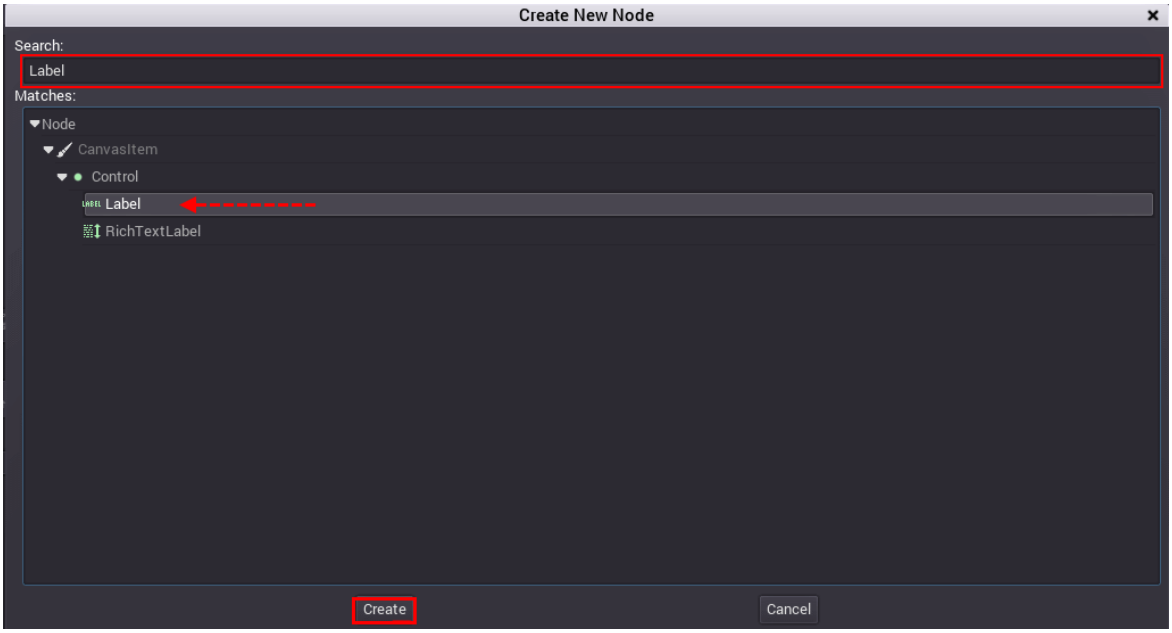



به صورت پیش فرض ، ویرایشگر Godot سه پنل مختلف دارد که در دو طرف پنل اصلی ( پنل میانی ) هستند : Scence ، File System ، Inspector ؛ در ادامه در مورد هر کدام از این پنل ها به تفصیل صحبت خواهیم کرد ؛ ولی فعلا می خواهیم یک گره به صحنه ی فعلی اضافه کنیم ، برای اینکار به به پنل Scene بروید و بر روی دکمه ی Add/Create a New Node کلیک کنید یا کلیدهای **Ctrl + A** صفحه کلید را بزنید تا وارد پنجره ی Create A New Node شوید .

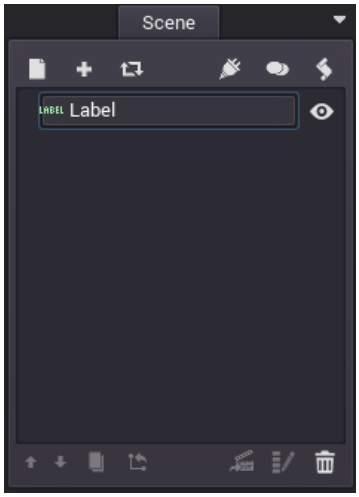



در این پنجره ی لیستی از گره های مختلفی که می توانید به پروژه اضافه کنید را می بینید .


وجود این گره های مختلف و پیش ساخته ، یکی از ویژگی های قدرتمند Godot است. فعلا می خواهیم فقط از گره ی ساده ی Lable را برای اضافه کردن متن به صحنه ، استفاده کنیم. برای اینکار ، در فیلد Search ، اسم گره مورد نظرمان را می نویسیم تا Godot به صورت خودکار این لیست بزرگ را فیلتر کند ( گره را جستجو کند ) :



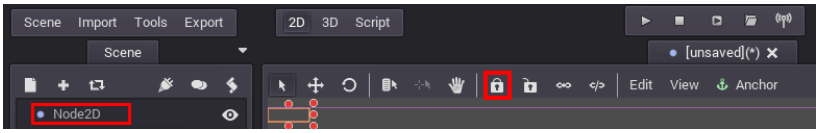

حالا با انتخاب گره Label بر روی Create بزنید تا گره به صحنه اضافه شود .



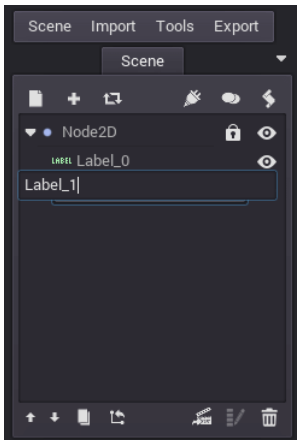

***توجه : در این آموزش به جای واژه ی Scene از « صحنه » و به جای واژه ی Node از « گره » استفاده می کنیم .***

حالا به گره دوم Label نیاز داریم ، برای اینکار یا می توانید مثل قبل عمل کنید و یک گره Label جدید بسازید یا می توانید از گره فعلی کپی بگیرید و اصطلاحا آن را Duplicated کنید . ( از آن یک « همتا / کلون » بسازید . از این به بعد به جای Duplicate کردن از واژه ی « همتا سازی » استفاده می کنیم ) . برای همتا سازی از گره موجود ، آن را انتخاب کنید و کلیدهای ترکیبی **Ctrl + D** را بر روی صفحه کلید فشار بدهید . پیغام خطای « This Operation can't be done on scene root » را به شما نشان می دهد که یعنی Godot نمی تواند این عملیات را بر روی گره ریشه ی صحنه انجام بدهد ، آن را تائید کنید. Godot بر خلاف سایر Game Engine های رایج از مفهومی به نام گره برای سازماندهی و مدل سازی اجزای بازی استفاده می کند . هر صحنه می تواند چندین گره به صورت **نمودار درختی** ( گراف درختی ) داشته باشد ؛ اگر ریاضیات دوران دبیرستان و یا دروس دانشگاهی را به یاد داشته باشید ، می دانید که ***هر درخت فقط یک « ریشه » دارد*** و سایر گره ها ، فرزندان گره ریشه هستند ، بنابراین به صورت منطقی نمی توانیم از گره ریشه در Godot ، همتا سازی کنیم . این مطلب را در جلسه ی بعدی به صورت کامل توضیح می دهم ؛ پس فعلا با انتخاب گره Label و کلیک بر روی آیکون سطل آشغال و یا زدن کلید Delete صفحه کلید آن را حذف کنید . حالا با زدن Ctrl + A و رفتن به پنجره ی Create A New Node به دنبال گره ی Node2D بگردید و آن را به صحنه اضافه کنید.

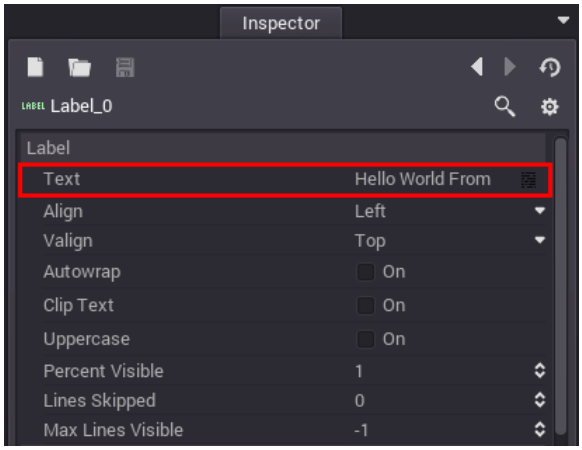

حال ، با انتخاب این گره بر روی آیکون قفل کلیک کنید تا دیگر مکان آن عوض نشود.



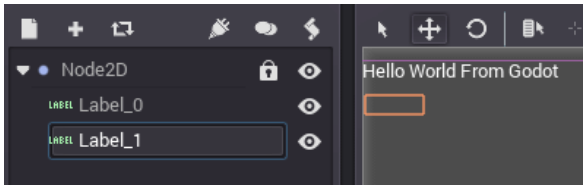

حالا گره Lable را به صحنه اضافه کنید و سپس با انتخاب آن و زدن روی Ctrl+D از آن یک همتا بسازید . می توانیم با انتخاب و کلیک کردن بر روی گره ها ، اسم آن ها را عوض کنیم . **توجه داشته باشید که در هنگام کُدنویسی باید به وسیله اسم گره ها ، به آن ها دسترسی پیدا کنیم** ، به همین دلیل بهتر است یک قاعده ی نامگذاری مناسب پیدا کنید و بر اساس آن ، نام هایی واضح و با معنا و مرتبط را برای گره های خود انتخاب کنید . در اینجا ، من اسم گره اول را Label_0 و اسم گره دوم را Label_1 می گذارم .




می توانیم با انتخاب گره مورد نظر Property ( ویژگی ، خصیصه ) های آن را در پنل Inspector ویرایش کنیم . مثلا با انتخاب گره Label_0 ، ویژگی Text آن را برابر عبارت « Hello World From Godot » می گذارم و برای اعمال آن کلید Enter را می زنم .

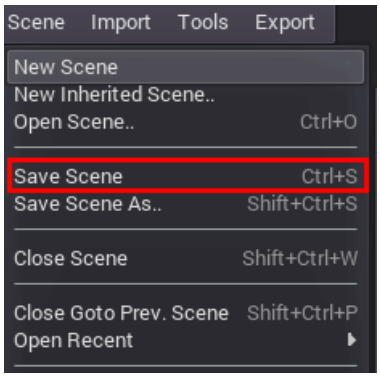



حالا با گرفتن گره دوم ، موقعیت آن را به زیر گره اول میاورم .

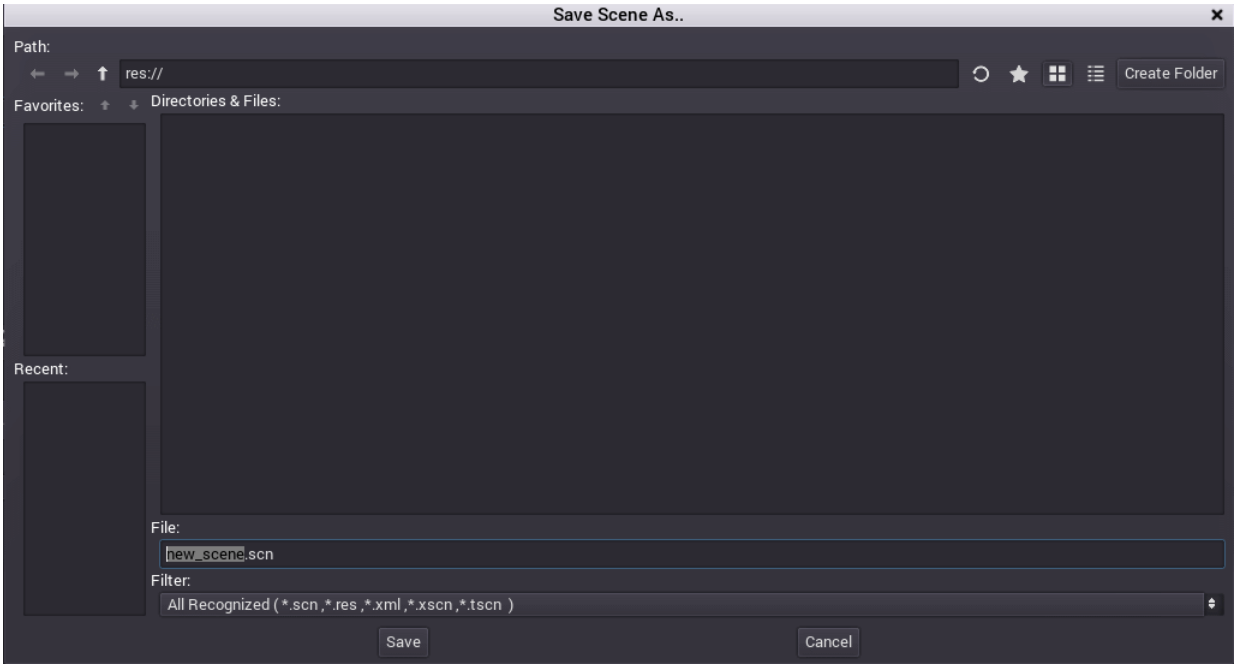



نکته ی مهم : پس از تغییر یا تعیین یک ویژگی ( Property ) در Inspector حتما باید کلید Enter را بزنید تا تغییرات اعمال شوند وگرنه ویرایشگر Godot تغییرات داده شده را نادیده می گیرد و آن ها را اعمال نمی کند

قبل از ادامه ی کار بهتر است این صحنه را ذخیره کنیم . برای این کار از منوی Scene بر روی گزینه ی Save Scene کلیک می کنیم و یا ، کلیدهای ترکیبی **Ctrl + S** را می زنیم

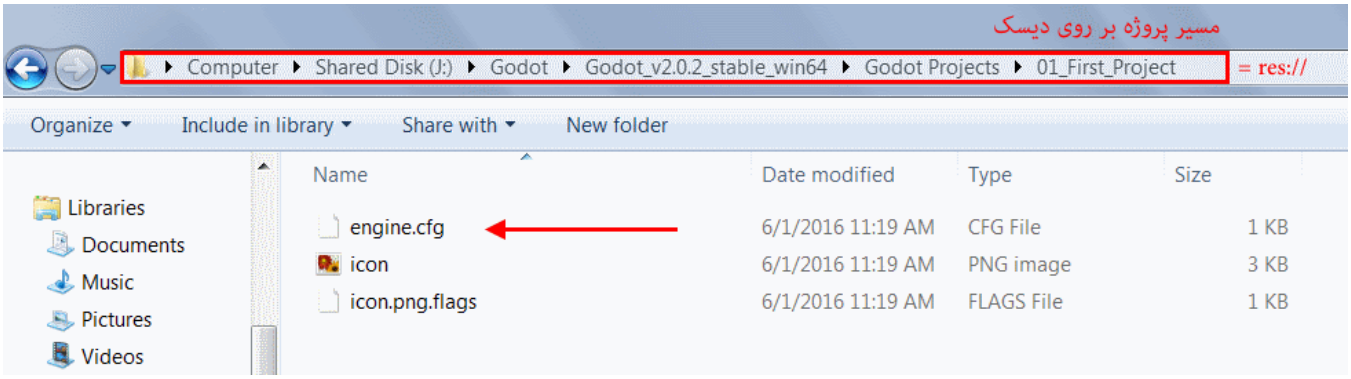



پنجره ی ... Save Scene As باز می شود که در آن می توانیم مسیر ذخیره سازی صحنه و نام صحنه را مشخص کنیم .

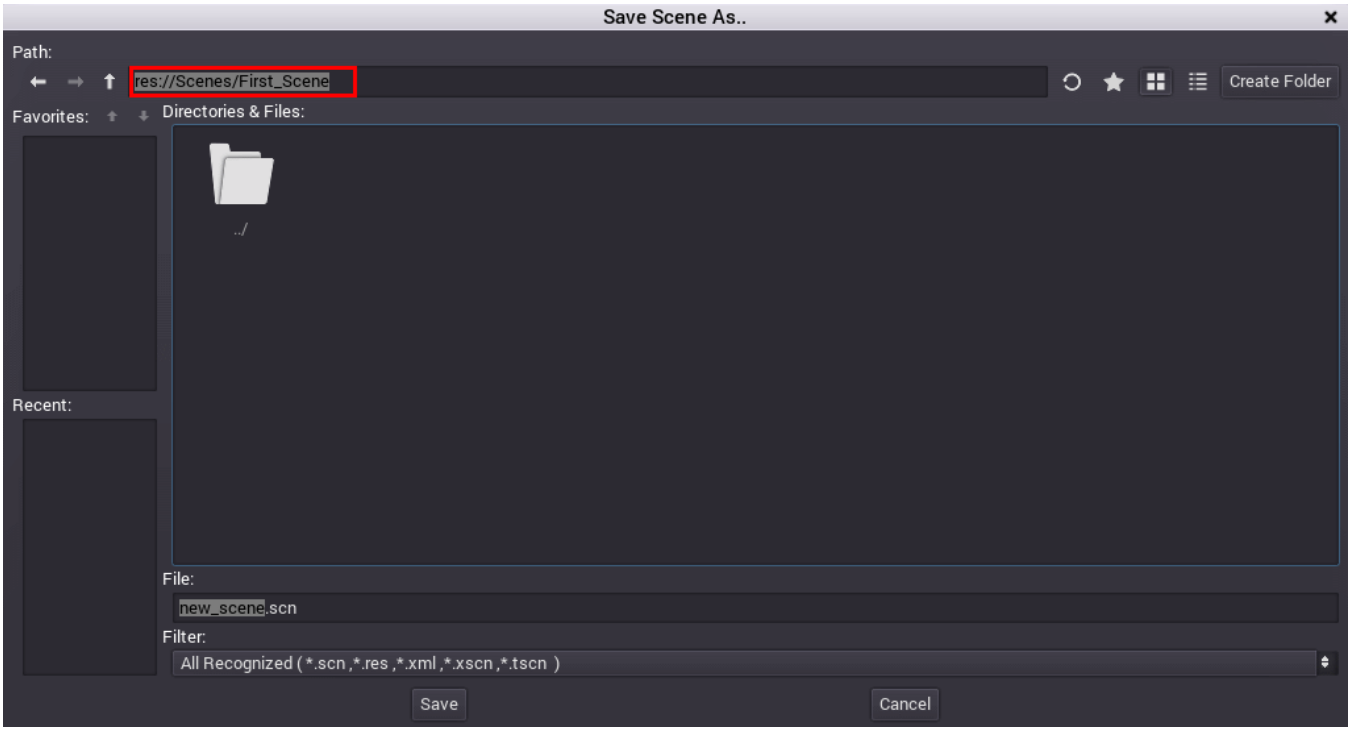



فیلد Path که مسیر ذخیره سازی پروژه را مشخص می کند ، اگر به آن نگاه کنید ، می بینید که با عبارت **res://** شروع می شود . این عبارت ، مسیر ریشه پروژه - پوشه ای را که در آن پروژه را ایجاد کردیم - را مشخص می کند که در اینجا پوشه ی 01_First_Project است . عبارت res:// معادل آدرس پوشه ای است که ، پروژه ما در آن قرار دارد ( این آدرس در کامپیوتر من به این شکل هست : J:\Godot\Godot_v2.0.2_stable_win64\Godot Projects\01_First_Project ) .

تمامی فایل های پروژه باید در مسیر ریشه ی پروژه قرار بگیرند ، Godot مسیر ریشه ی پروژه را مکان قرار گیری فایل engine.cfg ، در نظر می گیرد . بنابراین آدرس دهی به آن ها به صورت نسبی و از جایی که فایل اصلی engine.cfg در آن قرار دارد صورت می گیرد . در اینجا می توانید موقعیت پوشه و فایل های درون آن را در شرایط فعلی - قبل از ذخیره کردن چیزی - ببنید . ( به فایل engine.cfg دقت کنید . )




در Godot و در پنجره ی ... Save Scene As با زدن روی Create Folder ، ابتدا یک پوشه به نام Scenes ایجاد می کنم و داخل آن یک پوشه ی دیگر به نام First_Scene درست می کنم و وارد آن می شوم .




حالا اگر به فیلد Path نگاه کنید ، می بینید که آدرس این مسیر به این صورت است : **res://Scenes/First_Scene** ، ولی همین آدرس در ویندوز به این شکل است :

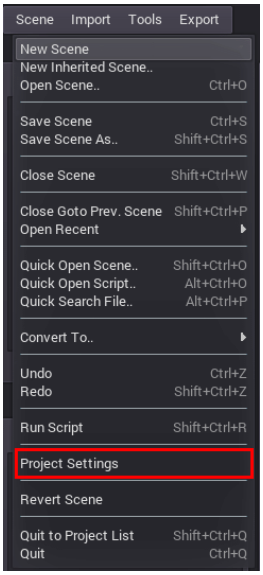
J:\Godot\Godot_v2.0.2_stable_win64\Godot Projects\01_First_Project\Scenes\First_Scene

*چون Godot یک Game Engine مولتی پلتفرم است ، از کاراکتر / - کاراکتر Slash به عنوان کاراکتر جداکننده ی پوشه ها در مسیر استفاده کرده است تا در هنگام ساخت بازی برای پلتفرم های مختلف ، مشکلی به وجود نیاید .*

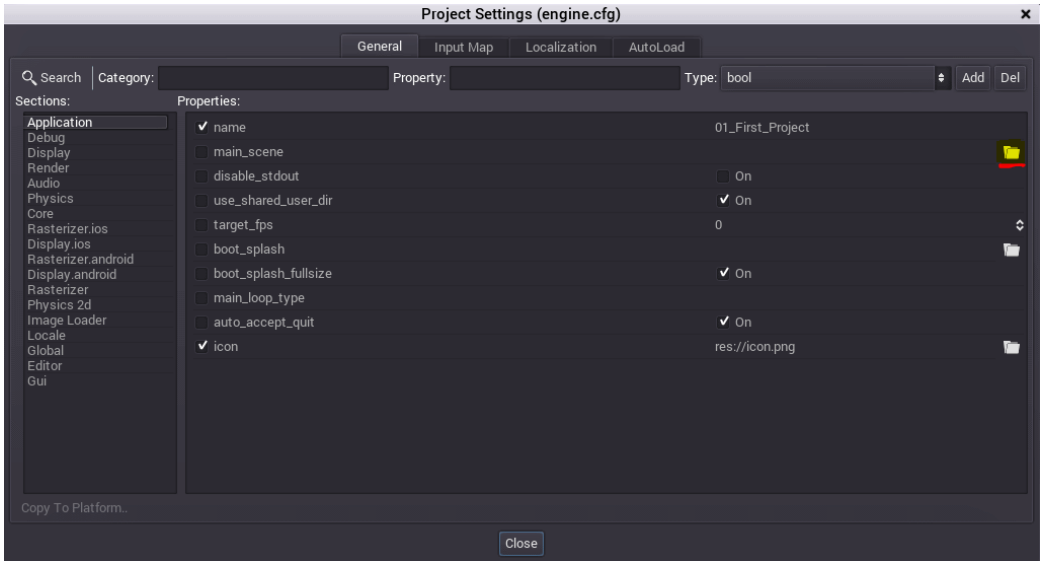
مسیر ذخیره سازی Scene را مشخص کردیم ، اسم First_Scene.scn را برای آن تعیین می کنیم . متاسفانه ، ویرایشگر Godot نمی تواند به صورت خودکار پسوند فایل ها را برای ما مشخص کند و به همین دلیل باید پسوند فایل ها را هم بنویسیم . ***پسوند صحنه ها در Godot برابر scn. است*** . پس از تعیین اسم برای صحنه روی Save می زنیم تا صحنه ذخیره شود .

حالا با زدن روی دکمه ی Play ، بازی رو اجرا می کنیم تا ببینیم چه شکلی شده . به محض زدن روی Play ، یک پیام هشدار به این مضمون به ما نشان داده می شود: « No main scene has ever defined » که یعنی « تا حالا هیچ صحنه ی اصلی تعریف نشده است .»

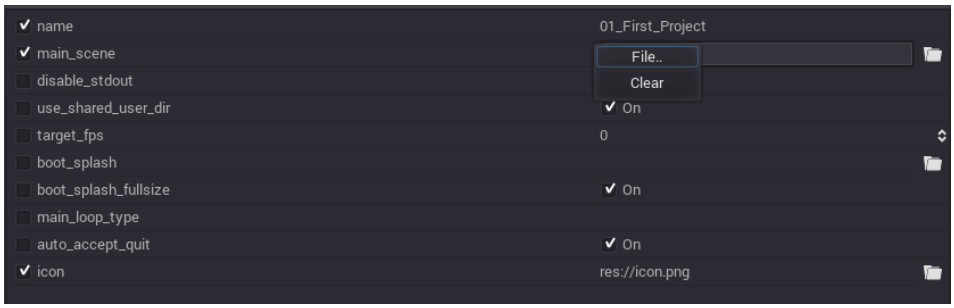

در Godot می توانیم صحنه های بسیار زیادی داشته باشیم ، مثلا یک صحنه برای استارت منو ، یک صحنه برای مرحله ی اول ، یک صحنه برای مرحله ی دوم و الی آخر ؛ به همین دلیل باید به صورت دقیق به Godot بگوییم که چه صحنه ای، صحنه ی اصلی و آغازین بازی ماست تا هنگام اجرا شدن بازی ، آن صحنه را اجرا کند . برای مشخص کردن صحنه ی اصلی ، به منوی Scene می رویم و با کلیک روی گزینه ی Project Settings ، وارد پنجره ی Project Settings (engine.cfg ) می شویم .



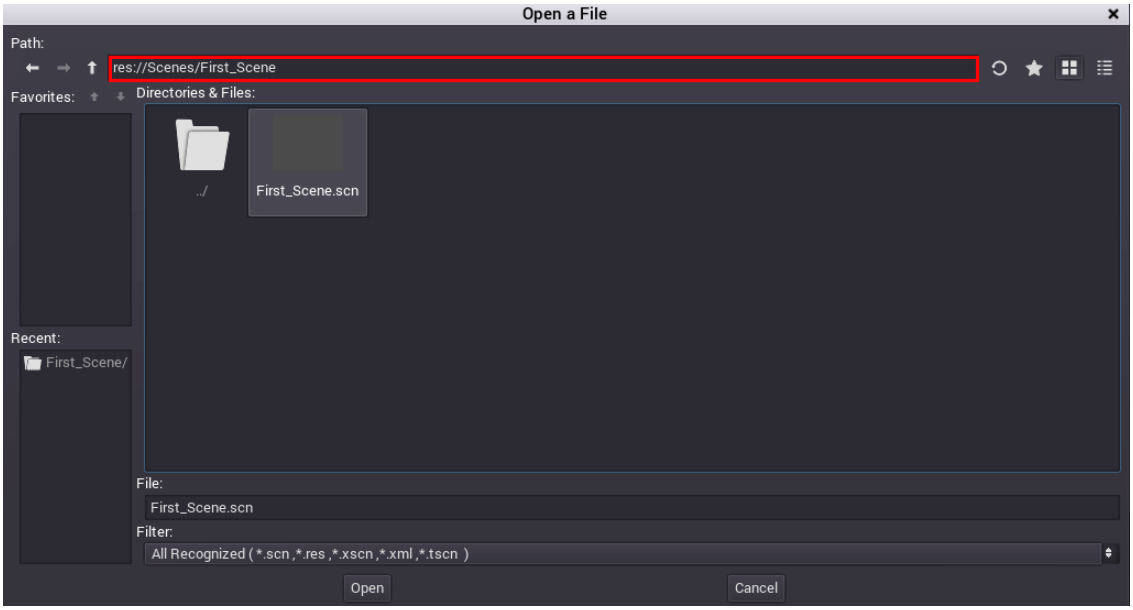

از این پنجره می توانیم مشخصات پروژه که در فایل egnie.cfg قرار دارند را تعیین و تنظیم کنیم ( یادآوری : Godot محل قرار گیری فایل engine.cfg به عنوان مسیر ریشه – پوشه ی ریشه – پروژه در نظر می گیرد . )




برای تعیین صحنه ی اصلی ، از تب General ، تیک گزینه ی main scene را می زنیم تا فعال شود ، سپس با کلیک روی آیکون پوشه و انتخاب گزینه ی فایل ، وارد Browser می شویم .

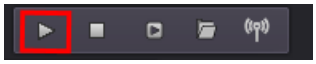



به محل مورد نظرمان می رویم و First_Scene.scn را انتخاب می کنیم و روی دکمه ی Open می زنیم .

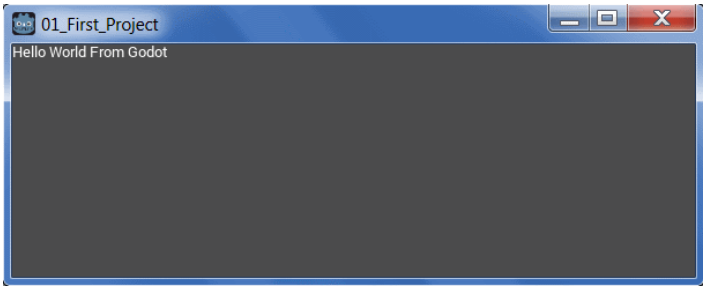



حالا پروژه می داند که صحنه ی اصلی در کدام مسیر است و چه نامی دارد .

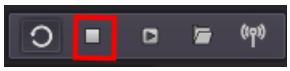

با کلیک روی دکمه ی Close به ویرایشگر بر می گردیم . کلیدهای ترکیبی Ctrl+ S را می زنیم تا صحنه را ذخیره کنیم و بعد و بر روی Play می زنیم تا بازی اجرا شود .



احتمالا فایر وال ویندوز یک پیام هشدار به شما نشان دهد ، یکی از گزینه های را انتخاب کنید و روی Allow Access بزنید تا بازی اجرا شود . می بینید متن Hello World From Godot در یک پنجره نشان داده می شود .



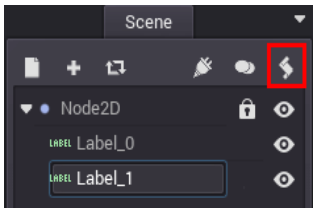

برای متوقف کردن بازی روی دکمه ی Stop کلیک می کنیم و با اینکار دوباره به ویرایشگر بر می گردیم.



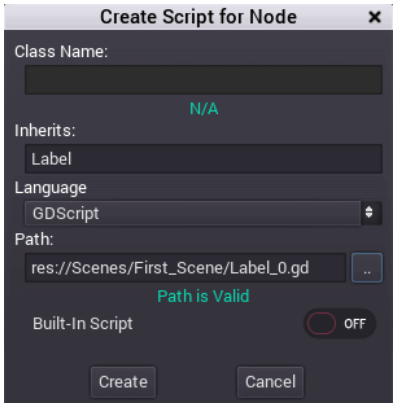
## اضافه کردن اسکریپت به گره :

متن گره Label_0 را به وسیله ی ویرایش ویژگی Text از داخل پنل Inspector تعیین کردیم ، ولی می خواهیم متن گره دوم را به وسیله ی کُد نویسی مشخص کنیم ،، برای اینکار نیازی به اسکریپت داریم . ***در Godot ، اسکریپت ها ، کُدهایی هستند که عملکرد و رفتار گره ها ، صحنه ها ، و به صورت کلی بازی را مشخص می کنند .*** در Godot ، تقریبا می توان به هر چیزی یک اسکریپت اضافه کرد و به وسیله ی اسکریپت رفتار گره را کنترل کرد !

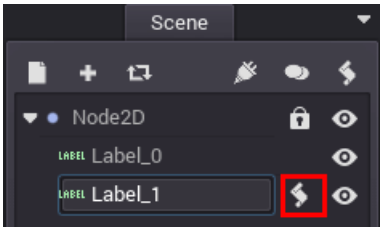
برای اضافه کردن اسکریپت به گره Label_1 ، با انتخاب آن در پنل Scene ، بر روی دکمه ی Edite/Create the Node Script کلیک می کنم .



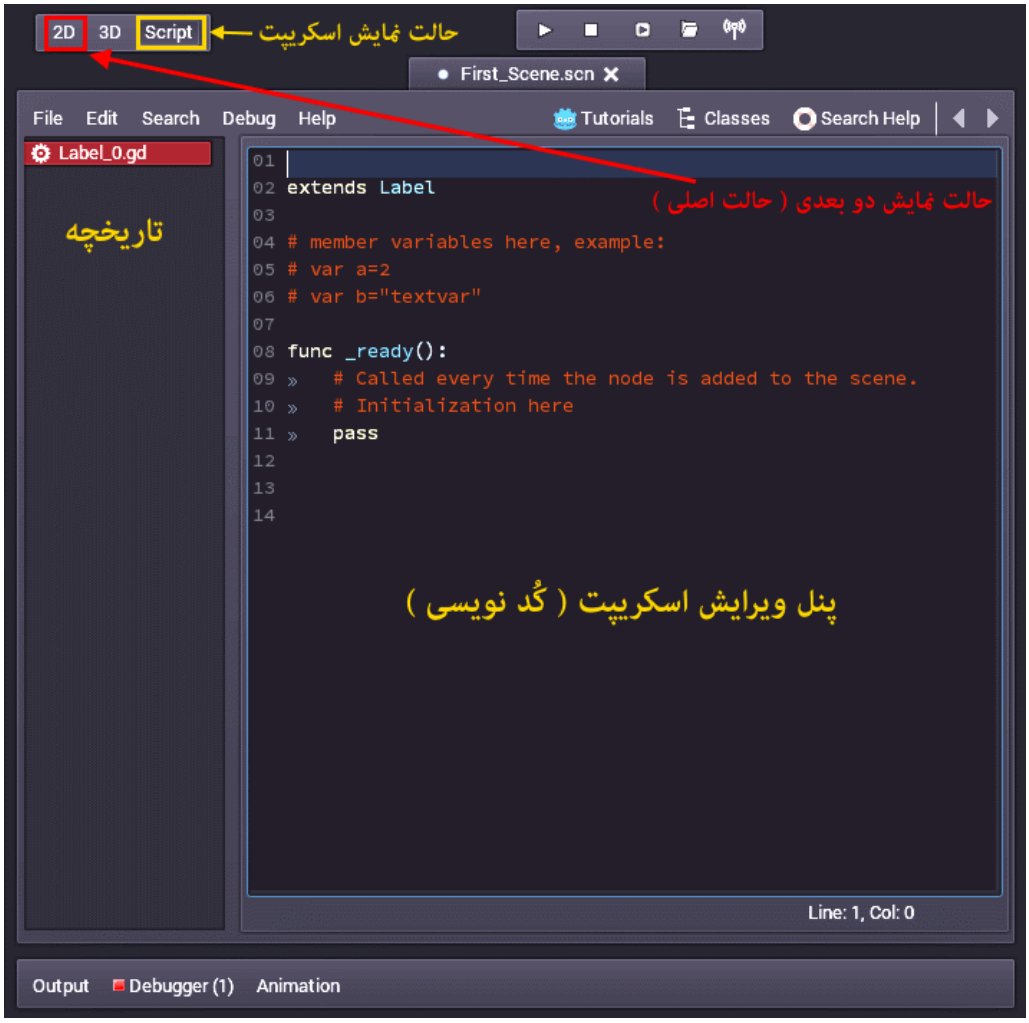

پنجره ی Create Script For Node بازی می شود ، مسیر اسکریپت را در پوشه ی First_Scene می گذارم و با تعیین اسم Label_0.gd ، و زدن روی Create این اسکریپت را می سازم . به یاد داشته باشید که ***پسوند اسکریپت ها در Godot برابر gd. است .***




پس از ساخت اسکریپت ، آیکون نشانگر اسکریپت در کنار نام گره ای که اسکریپت را ساختیم ، ظاهر می شود . اگر این آیکون کنار هر گره ای باشد ، یعنی به آن گره ، یک اسکریپت وصل شده است .




به محض ساخت یک اسکریپت ، بلافاصله وارد حالت ویرایش اسکریپت می شویم . در این حالت پنل سمت چپ ، تاریخچه است و پنل سمت راست ، محیطی است که در آن می توانیم کُدها را ویرایش کنیم.




همانطور که می بینید Godot یک اسکریپت جدید درست کرده است که با دستور **extends Label** ، از کلاس Label ارث بری کرده است . ***در Godot ، به صورت پیش فرض ، هر اسکریپت از کلاسی هم اسم با نوع گره ای که برای آن ساخته شده است ، ارث بری می کند ، و به این صورت به متدهای درونی تعریف شده برای آن گره ، دسترسی پیدا می کند.*** یعنی اگر در این مثال ، یک اسکریپت برای گره Node2D می ساختیم ، آن اسکریپت با دستور extends Node2D از کلاس Node2D ارث بری می کرد. نوشته هایی پس از علامت # قرار گرفته اند ، نوشته های مستند سازی هستند و به عنوان توضیحاتی اضافی برای برنامه نویسی کاربُرد دارند . Godot هنگام ساخت بازی ( کامپایل کردن کُد ها ) ، نوشته های مستند سازی را نادیده می گیرد . متد یا تابع **_ready()** یک متد ارجح پذیرتر هست ( Overridable functions ) که در هنگامی که این گره از بازی اجرا می شود ، فراخوانی و اجرا می گردد . اگر با اصول برنامه نویسی شی گرا آشنایی دارید ، به راحتی معنای عباراتی همچون ، « ارث بری » ، « مستند سازی » ، « متد های ارجح پذیر یا Overridable Functions » را درک می کنید .

نکته : دو واژه ی متد و تابع در برنامه نویسی تقریبا به یک معنا هستند . علاوه بر این به جای Overridable Function ، از عبارت « متد ارجح پذیرتر » استفاده می کنم .

خب ، می خواهیم یک متن به Label_1 انتساب دهیم ، برای همین کُد بالا را به این صورت تغییر می دهیم :

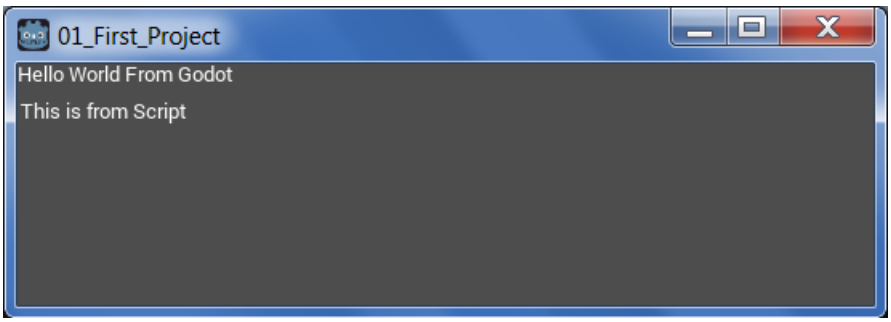
```
extends Label
func _ready():
        self.set_text("This is From Script")
```

سینتکس GDScript تقریبا مشابه زبان python است . در GDScript برای تعیین محدوده ی کُده از تورفتگی استفاده می شود. این تو رفتگی ها به وسیله ی زدن کلید Tab صفحه کلید ایجاد می شوند . توضیح کُد بالا به این صورت است

1. در خط اول با استفاده از کلمه ی کلیدی extends ، از کلاس Label ارث بری کردیم .
2. در خط دوم با استفاده از کلمه ی کلیدی **func** به برنامه گفتیم که می خواهیم یک تابع را ویرایش و یا تعریف کنیم ، سپس با نوشتن _ready() تابع سیستمی ( تابع درونی GDScript ) را مشخص کردیم که می خواهیم آن را Override کنیم ( کُدی را درون آن بنویسیم که نسبت به حالت پیش فرض این تابع ، ارجحیت بیشتری دارد ) و بعد با نوشتن علامت : شروع این تابع اعلان کردیم .
3. درون تابع با زدن کلید Tab ، نسبت به کلمه ی کلیدی func در خط بالا ، یک تو رفتگی ایجاد کردیم تا مشخص شود کُدهایی که می نویسیم متعلق به تابع func _ready(): هستند .
4. درون تابع با استفاده از کلمه ی کلیدی **self** مشخص می کنیم که داریم به گره فعلی ( همین گره ) اشاره می کنیم و سپس با استفاده از متد **set_text()** که متد درون Label است ، متن آن را می نویسم . متن باید درون دو علامت "" باشد .

حال با زدن کلیدهای Ctrl + S ، صحنه را ذخیره می کنیم . ***به صورت کلی بعد از اعمال تغییرات در کُد ، و قبل از اجرای بازی ، باید صحنه را ذخیره کرد .*** اگر بازی را اجرا کنیم ، چنین چیزی می بینیم .

تبریک می گم ! تونستید اولین پروژه ی خودتان را بسازید ، گره هایی به آن اضافه کنید ، از یک گره همتاسازی کنید ، ویژگی یک گره را در پنل Inspector عوض کنید ، صحنه را با نام صحیح ذخیره کنید ، صحنه ی اصلی را به پروژه معرفی کنید ، بازی را اجرا کنید و خروجی بگیرید ؛ اسکریپتی را به یک گره اضافه کنید ، در اسکریپت کُد نویسی کنید و نتیجه ی کُدی که نوشتید را ببینید !

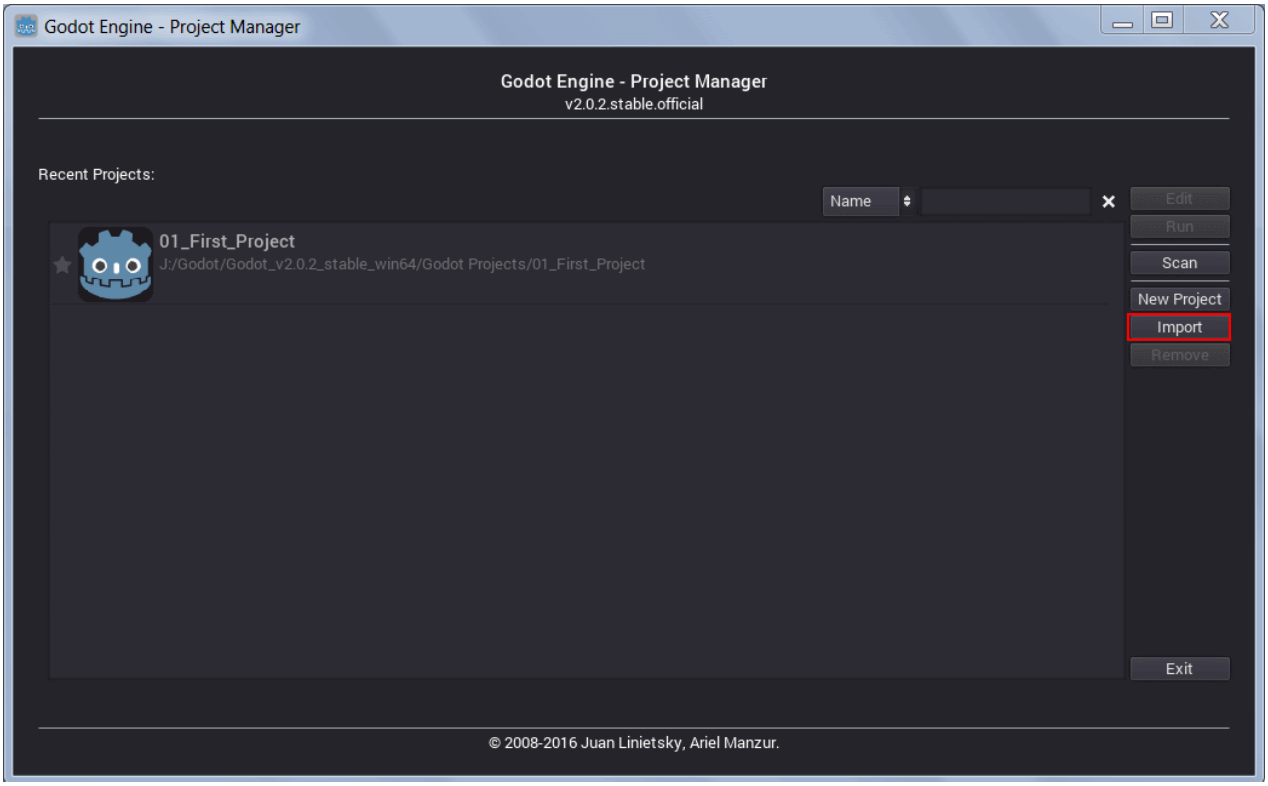



حالا برای خروج از ویرایشگر و رفتن به Project Manager ، از منوی Scene بر روی گزینه ی Quite to Project List کلیک کنید یا دکمه های **Shift + Ctrl + Q** را بزنید . یک پیام هشدار به این مضمون به شما نشان داده می شود که « باز کردن Project Manager ، تمام تغییرات ذخیره نشده از دست خواهند رفت » بر روی Yes کلیک کنید تا به Project Manager بروید.

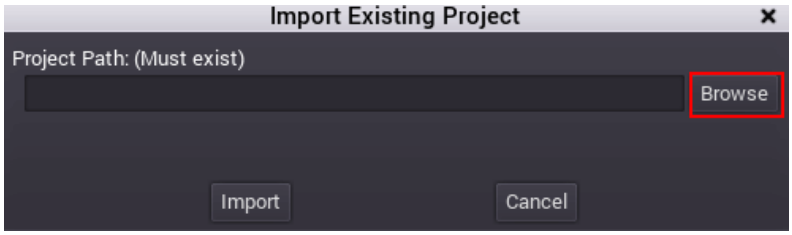
## وارد ( import ) کردن و ویرایش یک پروژه

تا اینجای کار با نحوه ی ساخت یک پروژه آشنا شدید . حالا می خواهیم نحوه ی وارد کردن یک پروژه ی موجود به ویرایشگر Godot را به شما نشان دهیم .

برای وارد کردن یک پروژه به Project Manager و ویرایش آن ، Project Manager را باز کرده و بر روی دکمه ی Import ، کلیک کنید .



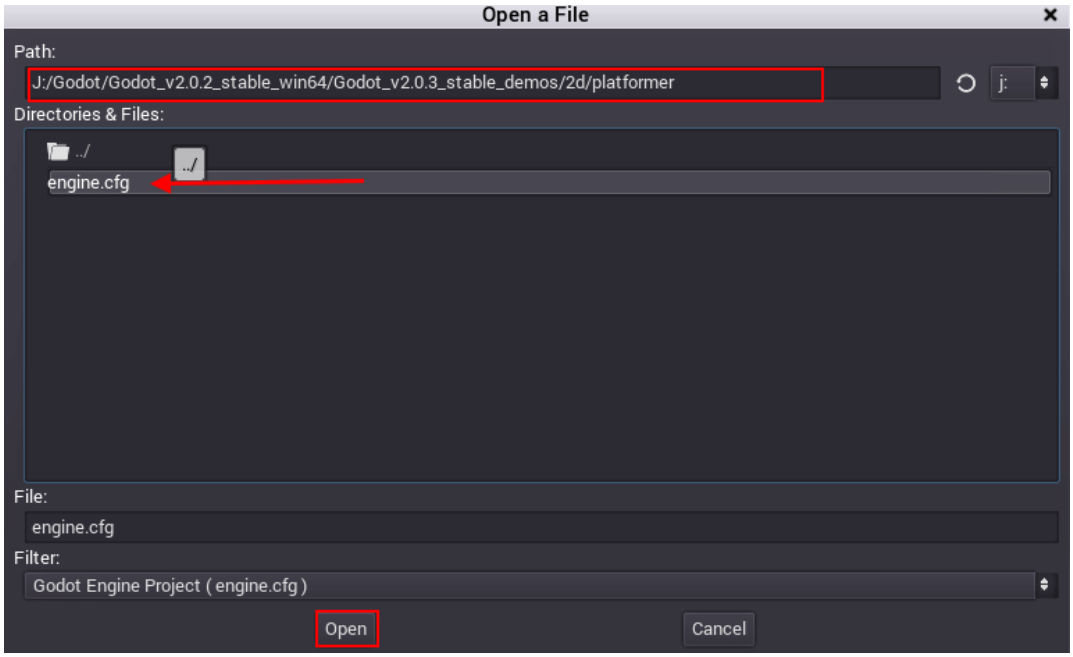

پنجره ی Import Existing Project باز می شود ، بر روی دکمه ی Browse کلیک کنید .



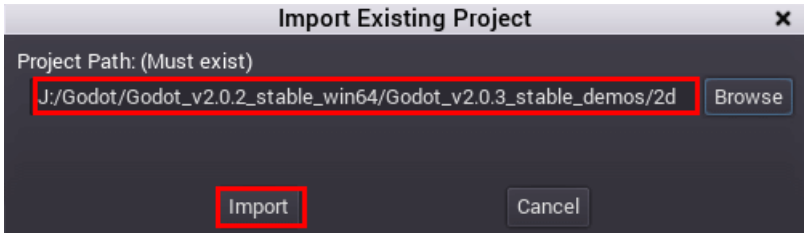

به پوشه ی ای که پروژه ی شما در آن قرار دارید بروید . در اینجا من می خواهم یکی از پروژه های تمرینی یا Demo به نام Platformer را که دانلود کرده ایم ، باز کنم ، این پروژه در این مسیر قرار دارد .

.../Godot_v2.0.3_stable_demos/2d/platformer

در این جا فایل engine.cfg را انتخاب کرده و دکمه ی Open را فشار دهید .

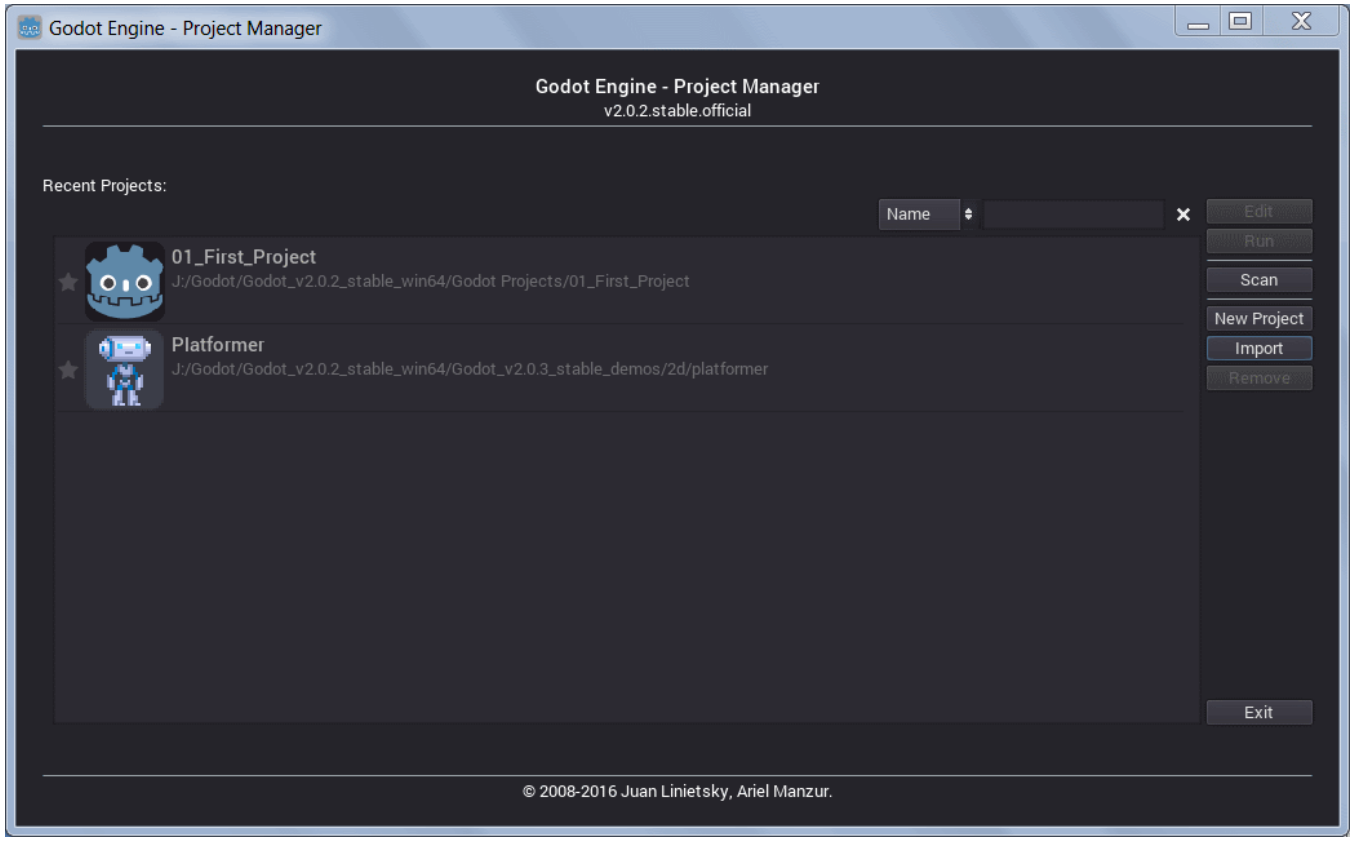



مسیر این فایل به صورت خودکار در فیلد Project Path ( Must Exist) اضافه می شود . بر روی دکمه ی Import کلیک کنید تا پروژه به لسیت پروژه های Project Manager اضافه شود .




حالا با انتخاب پروژه ی Platformer می توانیم کلیدهای زیر را فشار دهیم :

- ✓ Edit : پروژه را وارد ویرایشگر Godot می کند .
- ✓ Run : پروژه ( بازی ) را اجرا می کند .
- ✓ Remove : پروژه را از لیست پروژه های Project Manger حذف می کند ( فایل اصلی را از روی دیسک ، حذف نمی کند . )




ما با انتخاب پروژه و زدن کلید Edit وارد محیط ویرایشگر Godot می شویم .


خب ، به همین سادگی یک پروژه را وارد ویرایشگر Godot کردید و حالا می توانید آن را تغییر دهید . فایل های Demo و پروژه های تمرینی که دانلود کردیم ، یکی از بهترین منابع برای یادگیری کار با Godot هستند .

نوشته شده در تاریخ 1395/3/14

نسخه ی مورد استفاده Godot 2.0.2

نویسنده : رضا پویا