Programiranje v okolju Unity 3D

Rok Kos

11. januar 2016

Kazalo

Slike

Povzetek

V svoji projektni nalogi bom predstavil rokovanje z programskim okoljem Unity 3D. Kot primer bom vzel svojo aplikacijo, ki sem jo naredil z Unity-jem, ki predstavlja vodiča po Gimnaziji Vič.

Ključne besede: Unity3D

1 SET UP

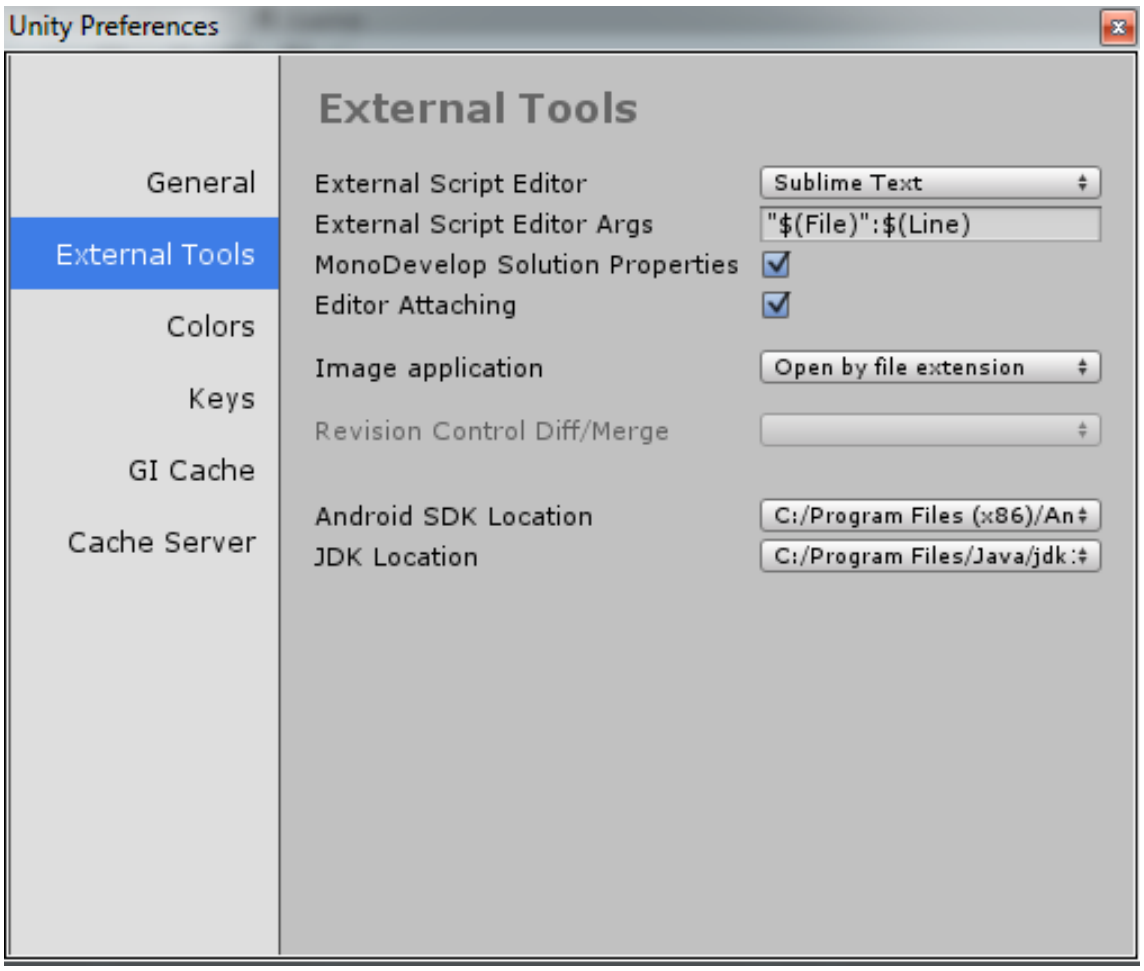
1.1 Editor

Kot vsako razvojno okolje ima tudi Unity 3D svoj privzeti editor. To je MonoDevelop, ki podpira večplatformski razvoj(to pomeni, da lahko isto kodo pišemo za različne operacijske sisteme Linux, Windows, Mac itd.). Podpira naslednje jezika:

- C#
- F#
- VisualBasic
- . . .

Ima tudi integrirano dopolnjevanje kode(code auto-completion) in svoj debuger. Sam sem kar nekaj časa uporabljal MonoDevelop in v njemu razvijal, ampak mi pri je pri njem vedno nekaj manjkalo. Zdelo se mi je, da je njihov code auto-completion vsiljeval svoje stvari in da mi highliter ni vedno obarval kakšnih klasov. Zato sem se odločil, da bom začel uporabljati meni ljubši urejevalnik besedila to je Sublime Text 3. Pri tem urejevalniku sem dobil veliko svobodo pri urejanju, iskanju, popravljanju ter pri samem urejanju urejevalnika. To sublime omogoča s svojim Package Controlom, ki se ga da inštalirati preko te strani: https://packagecontrol.io/installation. Preko Package Controla lahko inštaliramo dodatne plugine in snippete za sublime, ki izboljšajo samo delovanje tega. O tej temi bi se dalo še veliko govoriti zato jo bom pustil za drugič. Seveda moramo Unityju povedati, da naj svoje datoteke odpira z sublimom. To naredimo tako:

1. Inštaliramo Sublime Text 3/2 preko te strani
2. Gremo v **Edit → Preferences → External Tools**(kot lahko vidimo v spodnji sliki) in spremenimo na[urejevalnik
3. Spremenimo **External Script Editor Args** v "$(File)":$(Line) zato, da bo sublime skočil v vrstico, kjer je error
4. Sedaj bi nam moral Unity odpreti sublime, ko dvokliknemo na datoteko, ki je skripta
5. V sublimu bomo še spremenili, katere datoteke hočemo videti v projeknem drevesu. To naredimo tako, da gremo v **Project → Save Project As** in notri vtipkamo tole kodo:



Slika 1: Sublime Text Set Up

```
{
        "folders":
        [
                {
                        "path": "Assets/Scripts",
                        "file_exclude_patterns":
                        [
                                "*.dll",
                                "*.meta"
                        ]
                }
        ]
}
```

in potem shranimo datoteko. To nam scrije vse nepotrebne .meta in .dll datoteke

6. S Package Controlom na koncu še inštaliramo dodatne snippete, ki nam pomagajo pri auto-completion in barvanju kode. Vse potrebne package najdemo na tej strani http://wiki.unity3d.com/index.

php/Using_Sublime_Text_as_a_script_editor, kjer so tudi bolj podrobna navodila za celoten setup Sublime Text-a.

Kot zanimivost pa bi vam rad pokazal, kako enostavno se da narediti snippete v Sublime Text-u, ki se potem sprožijo ob določenem zaporedju tipk in nakoncu tabulatorja.

```
1
2  "folders":
3  [
4          {
5                  "path": "Assets/Scripts",
6                  "file_exclude_patterns":
7                  [
8                          "*.dll",
9                          "*.meta"
10                 ]
11         }
12 ]
13
```

1.2 Plugini

V Unity-ju si lahko pomagamo z različni skriptami, ki jih ustvarimo izven Unity-ja. Tem skriptam pravimo plugini. Te nam lahko pomagajo pri samem urejanju projekta, olajševanju in otomatiziranju nekaterih stvari ali pa nam dodeljuje kakšne funkcije za prav posebno platformo. Prvim pravimo Managed plugins, drugim pa Native plugins [Tec]. Managed plugine ponavadi pišemo v C#, saj uporavljajo samo knjižnico .NET. Te plugini se skompajlajo v dinamične knjižnice(dynamical linked library) oz. DLL, ki jih potem vključimo v projekt. Native plugini pa so napisani v C, C++, Objective-C in ostalih jezikih. To pomeni, da damo možnost Unity, da lahko npr. kliče kodo Java ali C++. Seveda je glavna prednost tega, da napišemo svoje knjižnice za določeno platformo npr. za IOS svojo in za Android svojo.

2 SHADERS

2.1 Definicija

Shaderji so skripte, ki nam povejo, kako naj se vsak piksel na zaslonu zrendera. To seveda ni samo od tega kako so napisani shaderji ampak tudi od tega kakšne materiale in teksture uporabljamo. V shaderjih so zapisani algoritmi, ki uporabljajo vektorje za svetlost in barvo in s tem povejo kako naj se obarva piksel. Z verzijo Unity 5 je prišel ven tudi njihov

Štandart Shader", ki je zelo uporaben in se lahko zelo prilagaja. Ampak v tem projektu bom predstavil, kako napisati lasten shader.

2.2 Lastnosti shaderjev

Tole je prikaz enostavnega shaderja [Jos]:

```
1  Shader "DT\Basic\SimpleShader"{
2          SubSHader{
3                  Tags = {"RenderType" = Opaque}
4                  CGPROGRAM
5                  #pragma surface surf Lambert
6                  struct Input{
7                          //(1.0, 1.0, 1.0, 1.0) R, G, B, A
8                          float4 color : COLOR;
9                  };
10                 void surf(Input IN, inout SurfaceOutput o){
11                         o.Albedo = 1;
12                 }
13                 ENDCG
14         }
15         Fallback "Diffuse"
16 }
```

3 CLASSES

4 PREFABS

5 IMPORTING

6 BUILDING PROJECT

7 GIT

[Tec] [Ala14] [Tho15]

Viri in literatura

[Ala14] THORN Alan. *Pro Unity Game Development with C#*. Vol. 1. izdaja. New York: Apress, 2014. ISBN: 9781430267461.

[Tho15] FINNEGAN Thomas. *Learning Unity Android Game Development*. Vol. 1. izdaja. Birmingham: Packt Publishing, 2015. ISBN: 9781784394691.

[Jos] KINNEY Joshua. *Introduction to Scripting Shaders in Unity*. URL: http://www.digitaltutors.com/tutorial/1438-Introduction-to-Scripting-Shaders-in-Unity (visited on 2015).

[Tec] Unity Technologies. *Unity Manual*. URL: http://docs.unity3d.com/Manual/index.html (visited on 2015).