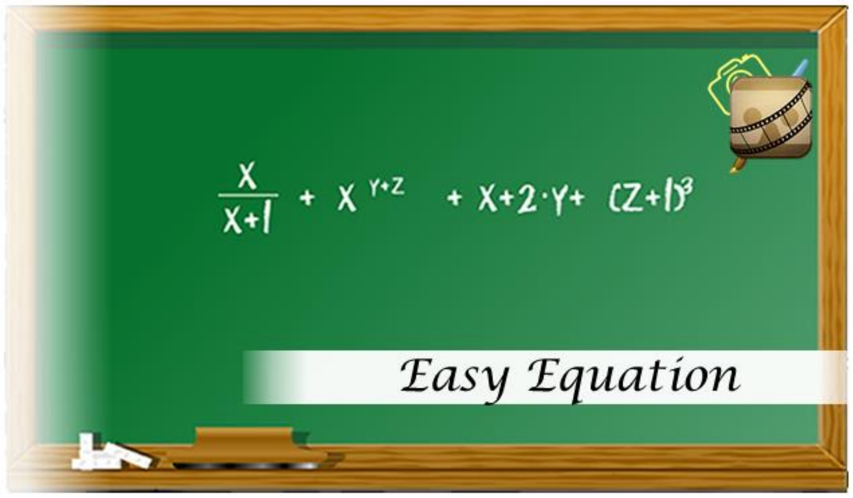
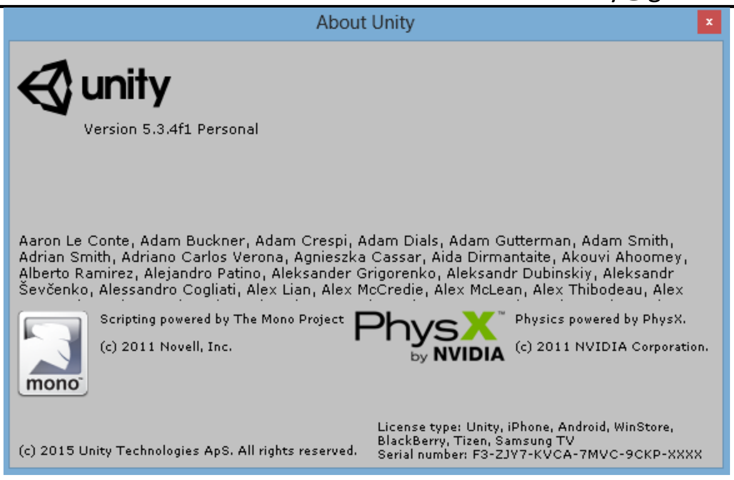
Simple Formula Generator



1. GENERAL INFORMATION

DATE OF DOCUMENT	29/06/2016
NAME OF THE PROJECT	Simple Formula Generator
AUTHOR	Michael Soler
UNITY VERSION	5.3.4.F1 PERSONAL
CONTACT	michael.soler.beatty@gmail.com


About Unity

unity
Version 5.3.4f1 Personal

Aaron Le Conte, Adam Buckner, Adam Crespi, Adam Dials, Adam Gutterman, Adam Smith, Adrian Smith, Adriano Carlos Verona, Agnieszka Cassar, Aida Dirmantaite, Akouvi Ahoomey, Alberto Ramirez, Alejandro Patino, Aleksander Grigorenko, Aleksandr Dubinskiy, Aleksandr Ševčenko, Alessandro Cogliati, Alex Lian, Alex McCredie, Alex McLean, Alex Thibodeau, Alex

Scripting powered by The Mono Project
(c) 2011 Novell, Inc.

Physics powered by PhysX.
(c) 2011 NVIDIA Corporation.

(c) 2015 Unity Technologies ApS. All rights reserved.

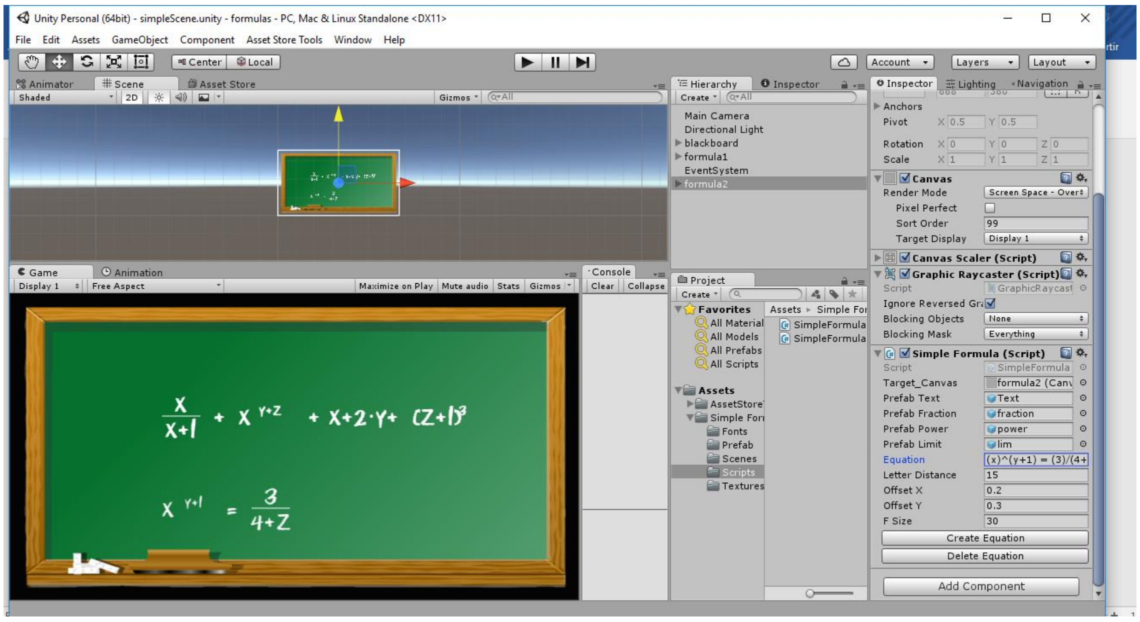
License type: Unity, iPhone, Android, WinStore, BlackBerry, Tizen, Samsung TV
Serial number: F3-ZJY7-KVCA-7MVC-9CKP-XXXX

Index

1. GENERAL INFORMATION 1
2. IMPORTING INFORMATION 2
3. PROJECT DESCRIPTION 3
4. LAYERS, TAGS AND COLLIDERS 3
5. SCRIPTING INFORMATION 3

2. IMPORTING INFORMATION

This package works independently and there is no need to import any other package.


Unity Personal (64bit) - simpleScene.unity - formulas - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Asset Store Tools Window Help
Hierarchy
Main Camera
Directional Light
blackboard
formula1
EventSystem
formula2
Project
Favorites
Assets
Create Equation
Delete Equation
Add Component

```
Pivot            X 0.5   Y 0.5
Rotation         X 0     Y 0     Z 0
Scale            X 1     Y 1     Z 1
Canvas
Render Mode      Screen Space - Over
Pixel Perfect
Sort Order       99
Target Display   Display 1
Canvas Scaler (Script)
Graphic Raycaster (Script)
Script           GraphicRaycast
Ignore Reversed Gr ✓
Blocking Objects None
Blocking Mask    Everything
Simple Formula (Script)
Script           SimpleFormula
Target_Canvas    formula2 (Canv
Prefab Text      Text
Prefab Fraction  fraction
Prefab Power     power
Prefab Limit     lim
Equation         (x)^(y+1) = (3)/(4+
Letter Distance  15
Offset X         0.2
Offset Y         0.3
F Size           30
```

Once the project is imported you could start using the demo scene to check how the equation script “SimpleFormula” works. Try changing the values and hitting the create/delete equation buttons.

3. PROJECT DESCRIPTION

This is a simple equation/formula generator that allows the user to write equations in text form into mathematical visual expressions as shown in the video. These expressions can be of the typeof exponential (a)^(b) ;   division (a)/(b) or limits (a)L(b). Only one level of indentation is possible with this package.

This package contains the following:

-The necessary textures, fonts and scene shown in the video.

-Scripts that manage the equation generation.

-Complete documentation to understand the principles of each package and full email support at: michael.soler.beatty@gmail.com.

4. LAYERS, TAGS AND COLLIDERS

LAYERS

- Default→ all objects are placed here.

TAGS:

- All objects are on the default layer.

COLLIDERS

- No colliders are used.

5. SCRIPTING INFORMATION

We always comment our script on the C# to make developers follow our code better. We have copied the main script's variables and functions in the following table:

SimpleFormula.cs	
This script is the one that manage the equations/formula of the asset.	
Important variables	**Important functions**
//This is the parent of the text objects that we are going to cre	//this function creates th
ate	e equation by adding/combi
public Canvas *target_Canvas*;	ning different prefabs and
/This is the prefab used for simple text	changing the texts inside
public GameObject *prefabText*;	public void *CreateEquation*
//This is the prefab used to create fractions of type ()/()	()
public GameObject *prefabFraction*;	
//This is the prefab used to create powers of type ()^()	
public GameObject *prefabPower*;	
/This is the prefab used to create limits of type ()L()	
public GameObject *prefabLimit*;	
// this is the equation in a text format	
public string *equation*;	
//this is the the public variable that is going to set the distan	

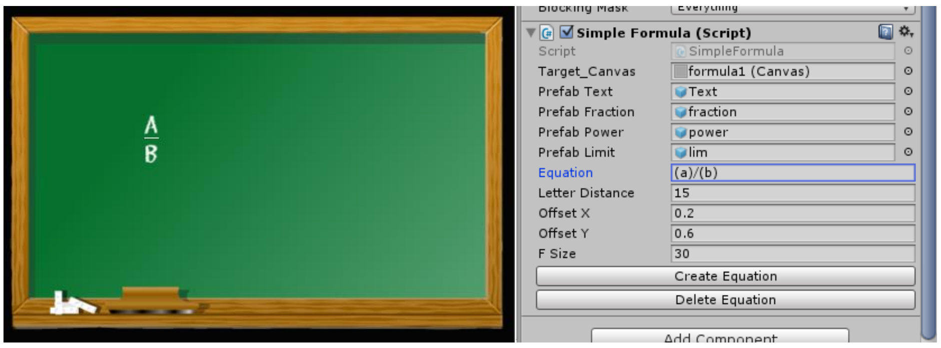
```
ces between equations:
public int letterDistance=10;
//offset to place X and Y text in percentage
public float offsetX, offsetY;
// size of the fonts:
public int fSize;

/these are local variables to refer to the instances of the formu
la types:
GameObject fraction, text, power, limit;
```

6. HOW THE PACKAGE WORKS

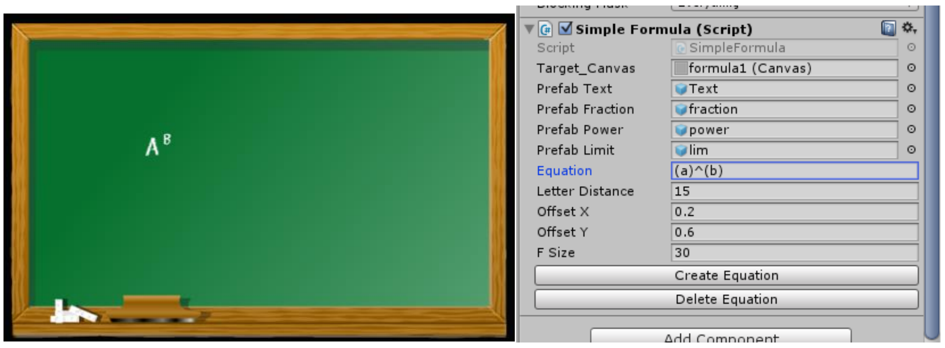
The package works as follows:

1. Write (a)/(b) to obtain $\frac{a}{b}$



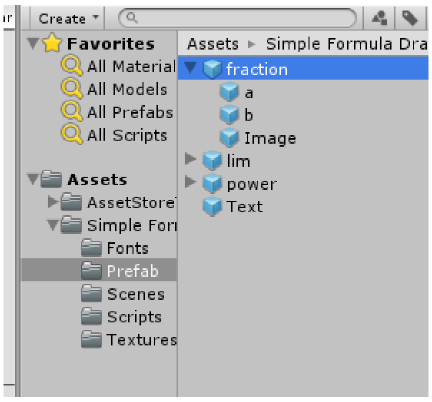
Script	SimpleFormula
Target_Canvas	formula1 (Canvas)
Prefab Text	Text
Prefab Fraction	fraction
Prefab Power	power
Prefab Limit	lim
Equation	(a)/(b)
Letter Distance	15
Offset X	0.2
Offset Y	0.6
F Size	30

2. Write (a)^(b) to obtain a^b

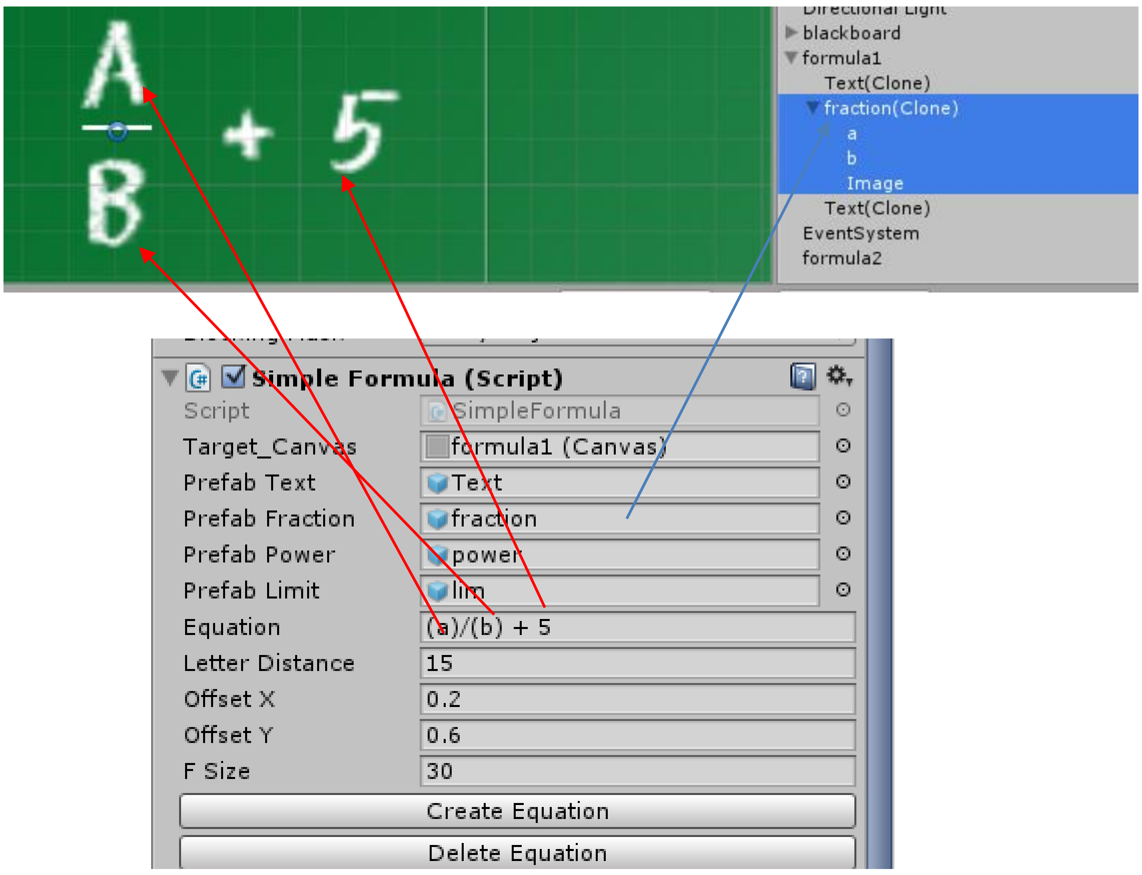


Script	SimpleFormula
Target_Canvas	formula1 (Canvas)
Prefab Text	Text
Prefab Fraction	fraction
Prefab Power	power
Prefab Limit	lim
Equation	(a)^(b)
Letter Distance	15
Offset X	0.2
Offset Y	0.6
F Size	30

These equations are achieved by using some prefabs that have text classes that are called and modified from script. Take for instance the case of the fraction:



When we write “(a)/(b) + 5” in the “SimpleFormula” script, the program is going to instanciate that gameobject in a specific location and it is going to add the variable values to the “a” and “b” text classes:



Script: SimpleFormula
Target_Canvas: formula1 (Canvas)
Prefab Text: Text
Prefab Fraction: fraction
Prefab Power: power
Prefab Limit: lim
Equation: (a)/(b) + 5
Letter Distance: 15
Offset X: 0.2
Offset Y: 0.6
F Size: 30

From the point of view of the code, it just compares the equation string to a known expected structure such as ()/() and perform specific actions once it is found:

```
// beginning of the ( a )
            if ("" + equation [jj] == "(") {

…
```

```
            // detecting the last part of the fraction
            if ("" + equation [jj] == "/") {

            // instantiate the gameobject
      fraction = Instantiate (prefabFraction, new Vector3 (0, 0, 0), Quaternion.Eu
ler (0, 0, 0)) as GameObject;
```

7. HOW THE CREATE MORE EXPRESSIONS

In order to create more mathematical expressions, you will need to create the prefab with a similar structure to the ones that already exist: two text components and other images if needed.

Then you will need to add a new symbol to the script in order to detect the new expression:

```
// detecting your own symbol
            if ("" + equation [jj] == "new symbol") {
```