Kings!

Card Swiping Decision Game Asset

V 1.20

Thank you for purchasing this asset!

If you encounter any errors / bugs, want to suggest new features/improvements or if anything is unclear (after you have read the documentation;) do not hesitate to contact us:

support@km-games.com

If you like our asset and want to support us, please leave a review at the Unity Asset Store for us. Thanks!

Contents

Getting started

- create a new Unity Project (2D) and import the Kings Game Asset
- open the „Game“-Scene in „Kings“-Project folder and press Play
- the game should now run
- this asset was built and tested with Unity 2017.1.1f1, if any error occurs we recommend to try it with this version of Unity
- read this documentation to get an overview of the features and how to use them

How to play

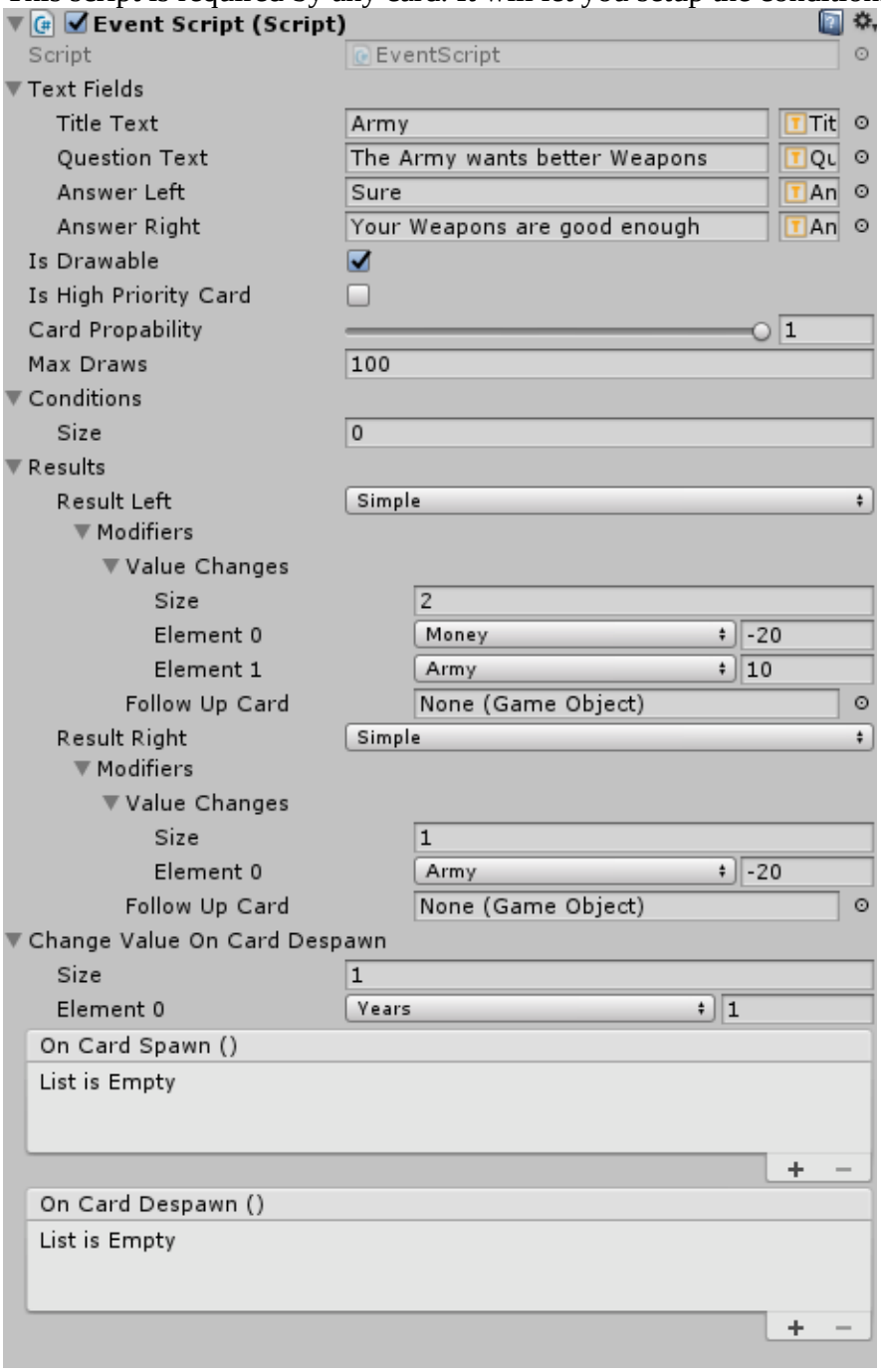
- in the game example you are the king (or queen) and have to make decisions by swiping left or right, to try to keep the 4 “factions”: Army, People, Religion and Money in Balance.
- if one faction reaches 0 you have lost and the game is over
- if you swipe up the game menu will open, you can see your “secondary stats”, these stats are randomly generate for every new game and will affect some of your decisions.
- in the game menu you will also find the achievements, highscore and settings menu, to close the game menu you simply have to swipe down

Modifying the Game

- we recommend you to edit the existing **Game** scene, this scene has been made to show all features of this asset and is kept as simple as possible for easy editing / re-skinning
- edit the Values in the **ValueDefinition** Script to your needs
- create your own cards, here we also recommend to duplicate one of the existing prefabs in cards folder, modify the look to your liking and use this one then as your base card for creating new ones

EventScript

This script is required by any card. It will let you setup the conditions and results of each card.



The **Text Fields** allow you to enter the Title, Questions and Answer for the card.

Is Drawable determines if the card can be randomly drawn from the CardStack. This should be always enabled, except for FollowUpCards.

Is High Priority Card will make sure the card is always drawn when the condition of the card is met.

Card Propability let's you allow to increase/decrease the probability that this card is drawn (when the condition is met). This doesn't affect High Priority Cards.

Max Draws let's you set up the value of how often this card can be drawn per game.

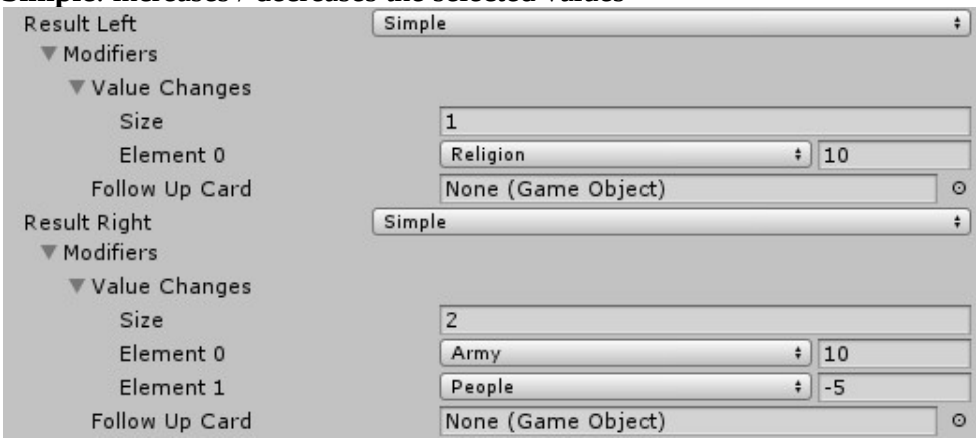
Conditions let's you specify when this card will be drawn. You can select as many conditions as you want. If you select no condition the card can be drawn any time (when the Sub Stack Condition of the Card Stack Group is met)

Results

Results are separated in two parts, **Result Left** and **Result Right** depending of the swipe direction.

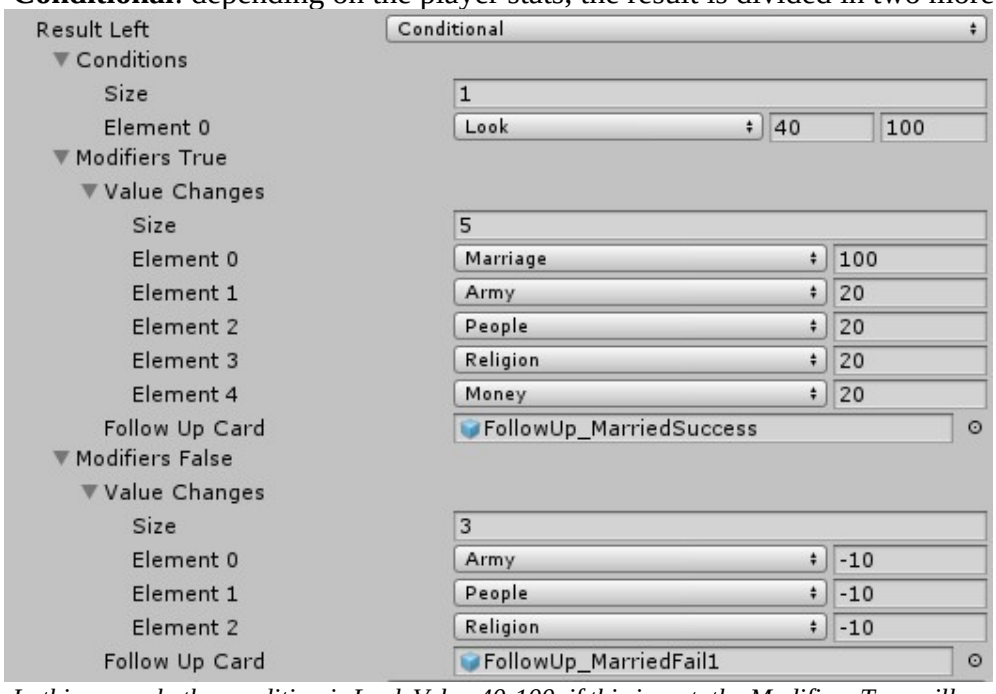
There are 4 different Results types: **Simple, Conditional, Random Conditions and Random**

Simple: increases / decreases the selected values



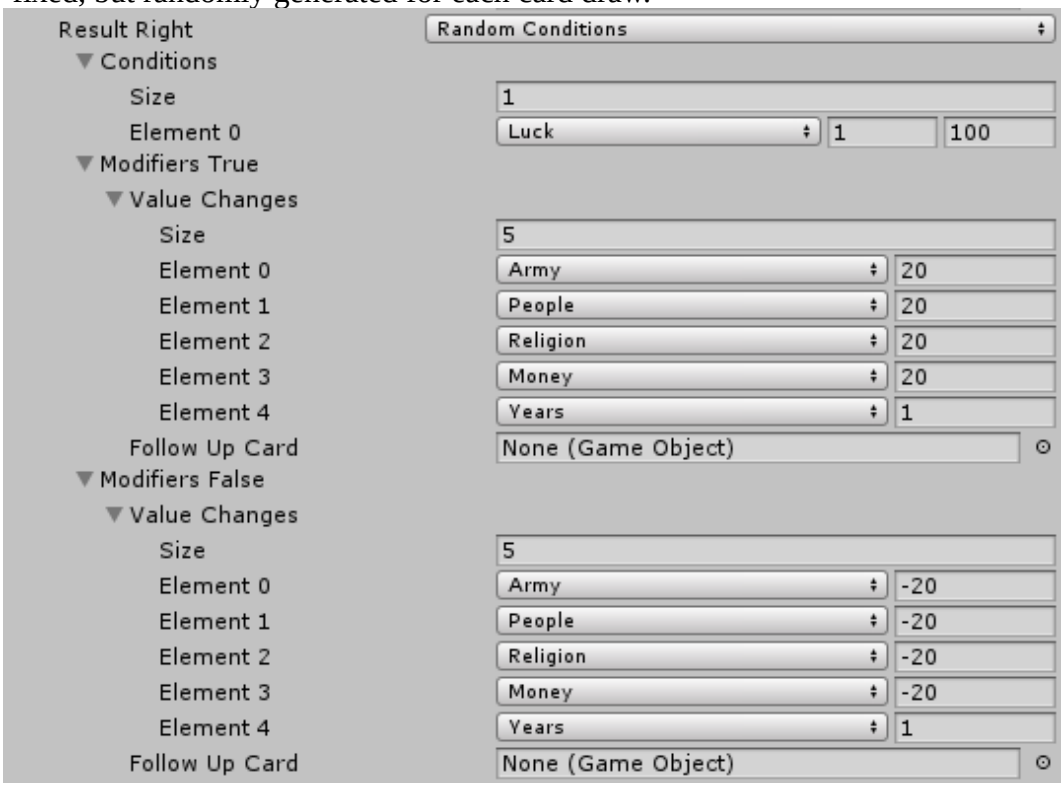
In this example left result will increase Religion Value by 10 and right result will increase Army value by 10 and reduce People value by 5.

Conditional: depending on the player stats, the result is divided in two more parts: true and false



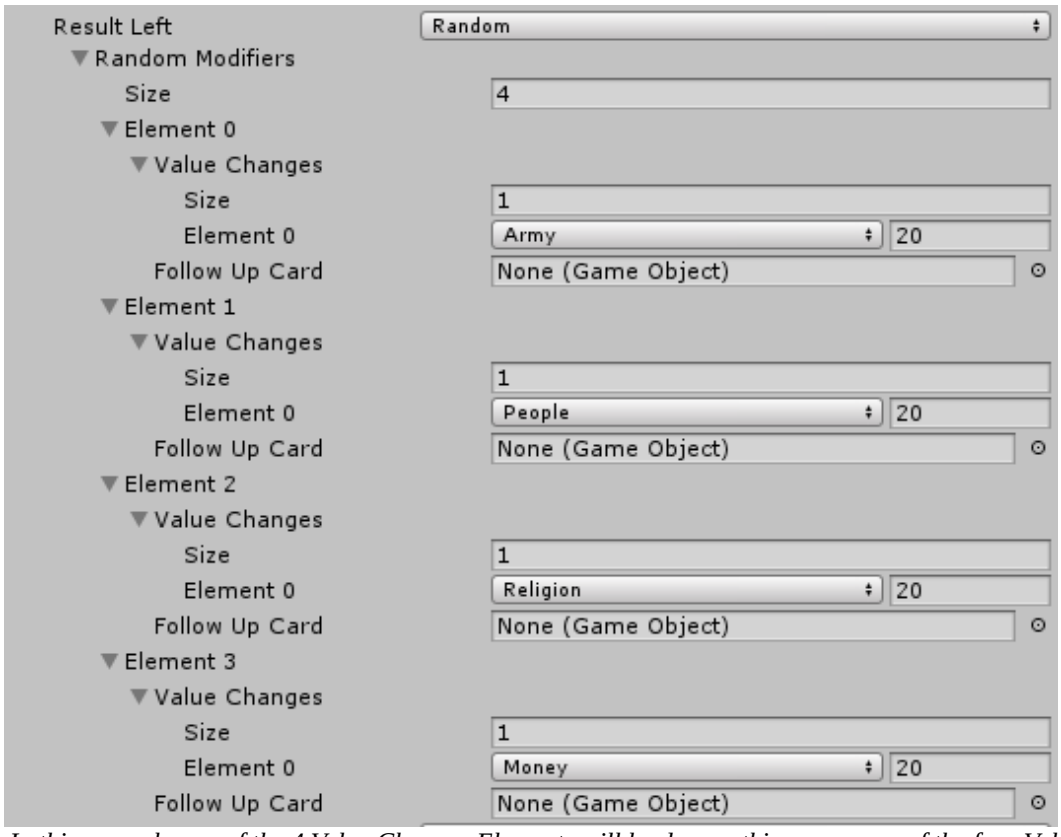
In this example the condition is Look Value 40-100, if this is met, the Modifiers True will apply, increasing the Marriage Value by 100 and Army, People, Religion and Money by 20, also a Follow Up Card "MarriedSuccess" is linked, this Card will be drawn directly after this one. If the condition is not met, e.g. Player has Look Value of 30, the Modifier False will be applied, descreasing the Army, People and Religion Values by 10, also the Follow Up Card "MarriedFail1" will be drawn.

Random Conditions: This is similar to the Conditional type, except the condition values are not fixed, but randomly generated for each card draw.



In this example the random condition is Luck Value, a random Value between 1-100 will be generated, e.g. 57, if the Luck Value of the Player is above this value the Modifiers True will apply, if it is below this value the Modifiers False will apply.

Random: one random element will be picked from a list of Value Changes



In this example one of the 4 Value Changes Elements will be drawn, this means one of the four Values Army, People, Religion or Money will be increased by Value 20.

Change Value on Card Despawn this is used if you want to change a value, regardless of which direction the player swiped, for example to increase the value Year by 1.

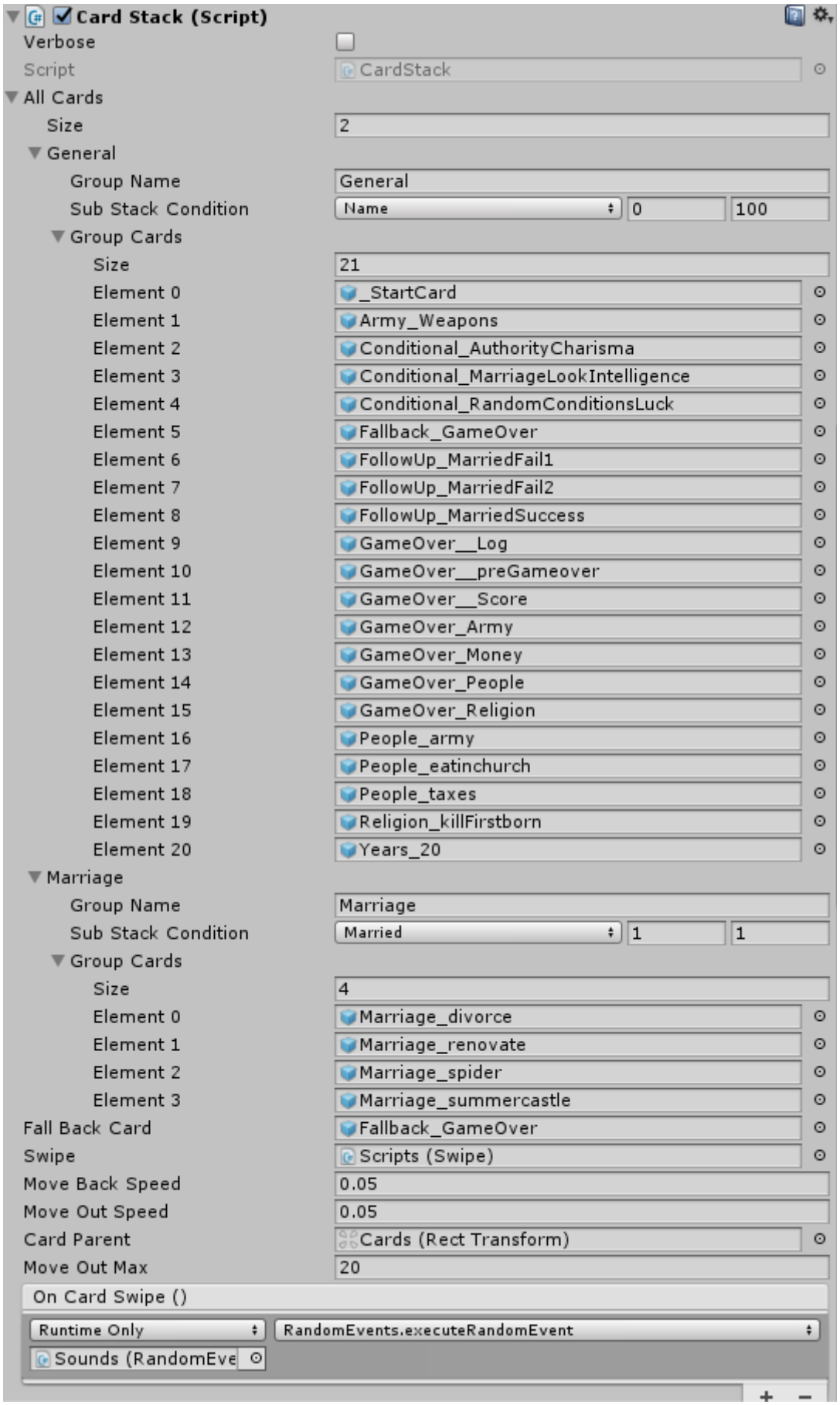
On Card Spawn Event, can be used to trigger a Unity Event each time the card is spawned. You can use this for example if you want to trigger an achievement when this card is drawn.

On Card Despawn Event, can be used to trigger a Unity Event each time the card despawns. This is the recommend way to add Game logs, Achievement points etc...

Since **On Card Spawn** can be done multiple times, if the player quiets the game and the active card has a **On Card Spawn** Event, which already triggered since the card is spawned, it will trigger again when the player resumes the game, because the card stack will automatically draw the last active card.

Card Stack (in Scripts)

This script manages which card will be drawn. All cards need to be linked into this script.



If you check the **Verbose** toggle you can display additional information in Play Mode which can be useful for debugging.

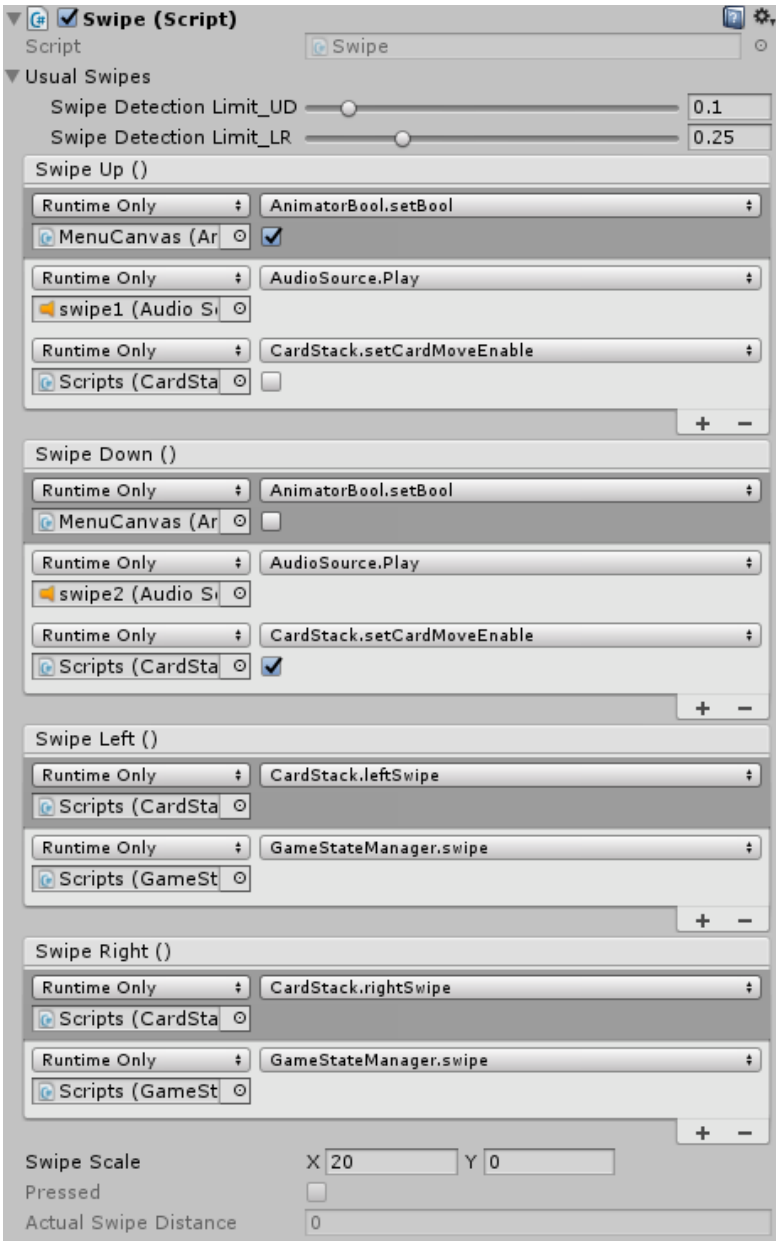
All Cards Size enter the number of how many different groups you want. In the above example we have two groups: “General” and “Marriage”.

In **Sub Stack Condition** you can choose a condition which is applied for this group and all cards in it, in this example the “Marriage” group has condition “Married Value 1-1” this means that the cards in this group can only be drawn if the player is married (The Value is 0 if the player is not married). For the “General” group you should select a value that is always there with a range that will always be met, e.g. 0-100. Update 1.20: You can now have more than one Sub Stack Condition.

Fall Back Card is used only when there is no more available card to draw (no card meets the current conditions). This should not happen and is mainly here only for preventing errors.

On Card Swipe lets you trigger an Event every time a card is swiped out of the screen.

Swipe Script (in Scripts)



Swipe Up sets the AnimatorBool of the MenuCanvas to true, to show the game menu. Plays a audio file (swipe1). And disables CardMove in CardStack script, to prevent swiping card while in the game menu.

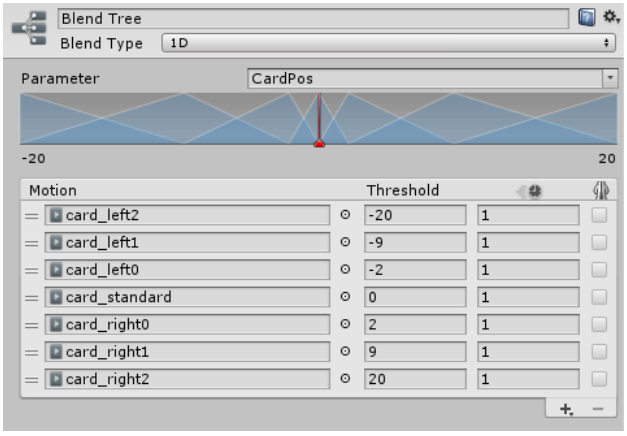
Swipe Down sets the AnimatorBool of the MenuCanvas to false, to hide the game menu. Plays a audio file (swipe2). And enables CardMove in CardStack script, to enable swiping cards again.

Swipe Left tells the CardStack to do a left swipe and the GameStateManager that there has been a swipe.

Swipe Right tells the CardStack to do a right swipe and the GameStateManager that there has been a swipe.

This script handles all swipe gestures.
Swiped Detection Limit_UD lets you set the value of how long (distance) a swipe has to be, to be recognized as a swipe for up or down swipes. **Swipe Detection Limit_LR** is the same just for left and right swipes.

Swipe Scale lets you scale the factor of your swipes for the Animator BlendTree to swipe the cards to the left or right.



Card Animator

Every card has the same Animator with this blend tree, depending on the swipe distance and swipe scale the appropriate animation will be displayed.

Value Definitions (in Project folder scripts)

This Script is not in the Hierarchy window of the game project.
You have to open it from the Project folder **Kings\scripts\valueDefinitions.cs**

valueDefinitions.cs

No selection

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class valueDefinitions : MonoBehaviour {
6
7      /*
8       * Define all your values of the game here.
9       * The value definitions are used in several scripts to
10      * show easy dropdown menues e.g. for conditions or value modifications.
11      */
12
13     public enum values{
14         name,
15         surname,
16         gender,
17         country,
18         years,
19         army,
20         people,
21         religion,
22         money,
23         authority,
24         intelligence,
25         charisma,
26         luck,
27         creativity,
28         look,
29         health,
30         marriage,
31         married
32     }
33 }
34
```

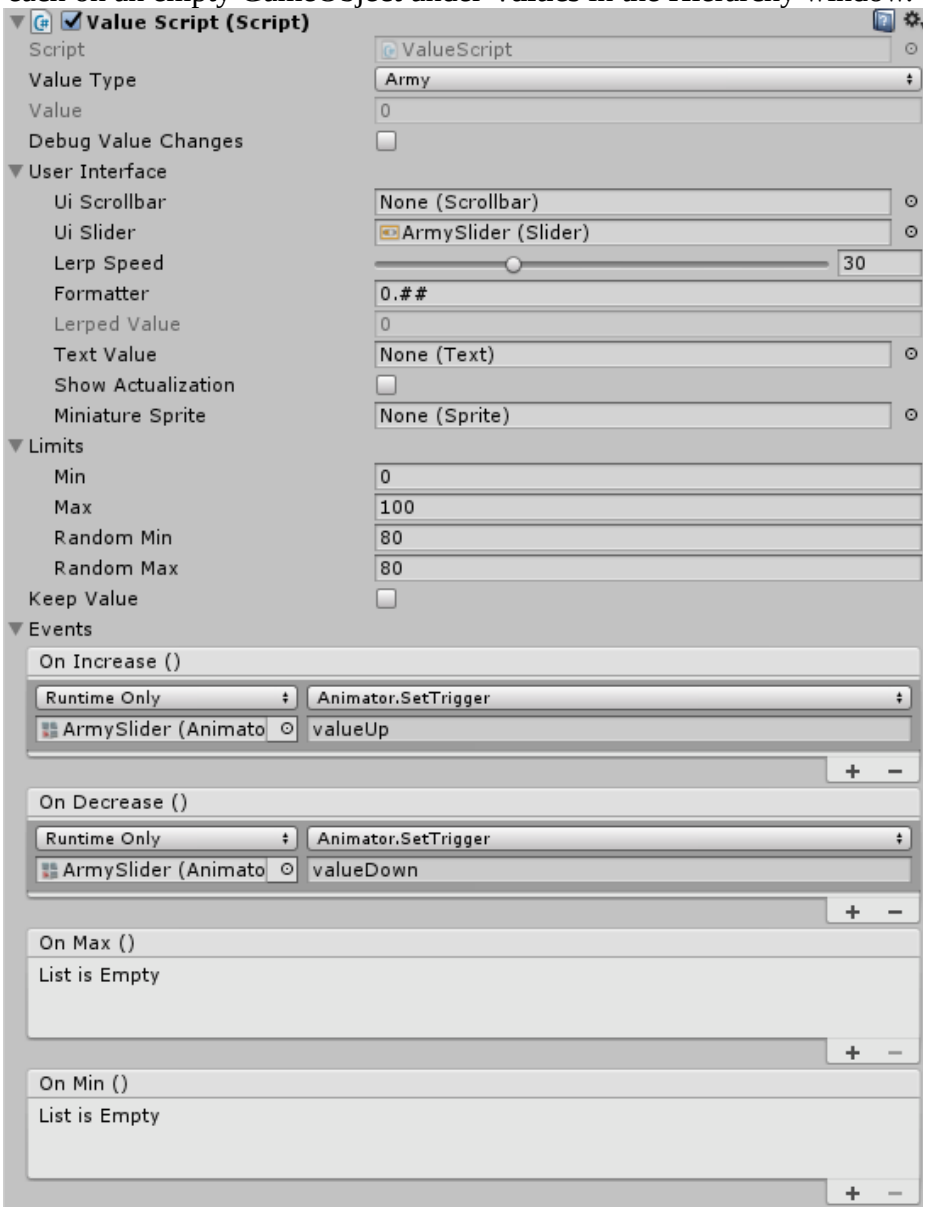
Every Value needs to be listed in this **s**cript, you need to open this script and add/edit the values here. These values will be used in many other scripts like EventScript, where you can simply select them from a drop-down menu.

PLEASE NOTE: If you remove a value from this list (remove line) it will reorder all elements and mess up all existing cards, since these values are addressed by there line number.

When you start your game project the first thing for you to do should be creating your list with all values needed. Adding additional values at the bottom or editing/renaming existing values can be done without a problem.

Value Script (in Values)

For every value you will also need a Value Script. In the example Game scene they are attached each on an empty GameObject under **Values** in the Hierarchy window.



In **Value Type**, you simply choose from the drop down menu for which value this Value Script is.

If you want to display this value with a slider you can link it in **UI Slider**, if you want to display it as a text value, you can link it in **Text Value**, of course you can also do both or none if you dont want the value to be shown.

If you enable the **Show Actualization Toggle** a popup window (UpdatedStatsPanel) will appear every time this value changes. This is used if you want to show the player that a value of a secondary stat (StatsPanel, when you swipe up) has changed. You should also then link in a **Miniature Sprite** which will be displayed then.

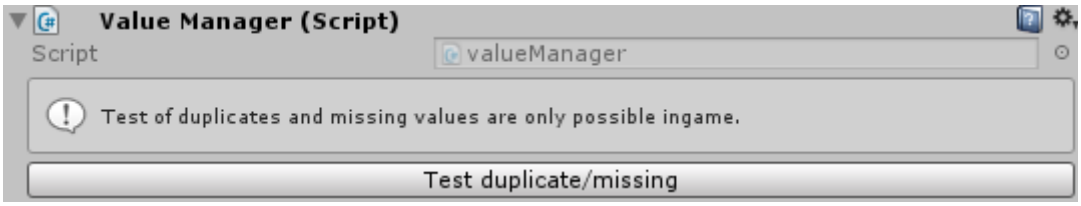
Limits let you set the **Min** and **Max** range for the value, in this example from 0-100, this means this value can not get lower than 0 or higher than 100.

The **Random Min Max** values are the values which are generated on every new game start, in this example the range is from 80-80, this means the value can only be 80. If you choose for example 10-50, it will generate a random value between 10 and 50.

The Events for **On Increase, On Decrease, On Min, On Max,** can be used for example if you want to animate the stats display.

New in Update 1.20:
Keep Value allows persistent values which are not reset on a new game, this can be useful if you want to keep the years value always growing.

Value Manager (in Values)

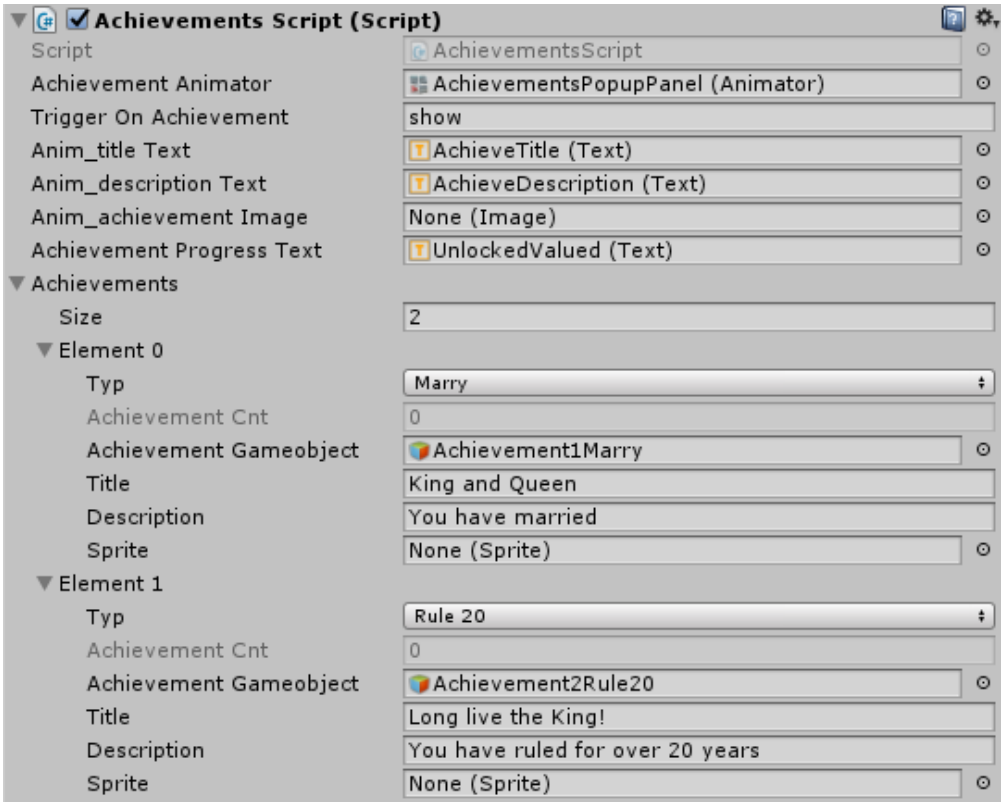


When in Play Mode you can run a **Test for duplicate/missing values**. For example if you set up a value in the Value Definitions Script but did not create a GameObject with the Value Script for it, or if you created two GameObjects for the same value.

Value Dependent Enable Script (in Values)

With this script you can display a specific GameObject (Slider, Text Value, etc.) only when the Value Limit is reached. In this example the Marriage Slider will only be displayed when the Marriage Value is above 1.

Achievements (in Scripts)

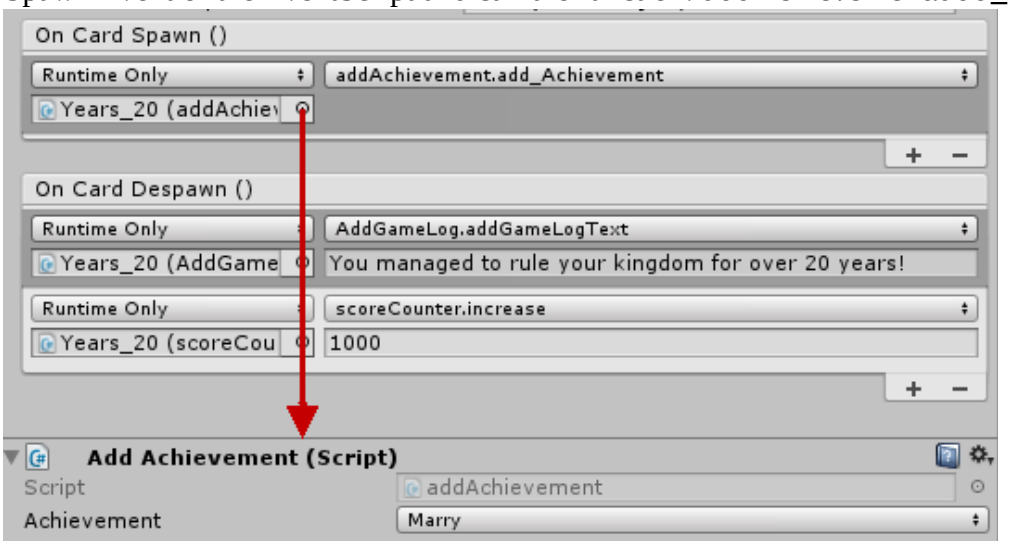


This is a simple Achievement Manager to popup Achievements on special Events. For every Achievement you have you need to set up manually the list in the Achievement Script, similar like the Value Definitions.

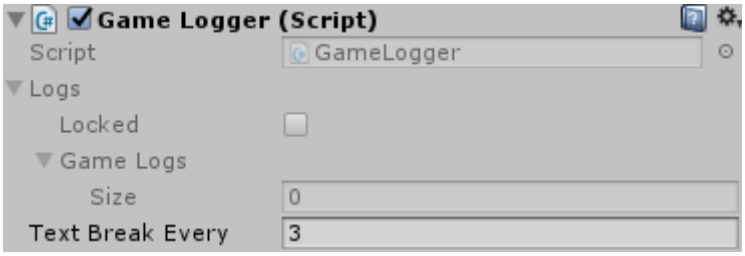
```
public enum achievementTyp{
    marry,
    rule20
}
```

To display the unlocked Achievements in the Achievements Screen you have to create a GameObject (with title, description, image etc.) and link it in **Achievement Gameobject.** Two example Achievements are created you can find them in :
MenuCanvas → Panels → AchievementsPanel → Scroll View → Content

To trigger an Achievement simply add the **Add Achievement Script** to the card which should trigger it, select the appropriate Achievement from the drop-down menu and link it in the On Card Spawn Event of the EventScript and call the function: **addAchievement.add_Achievement.**



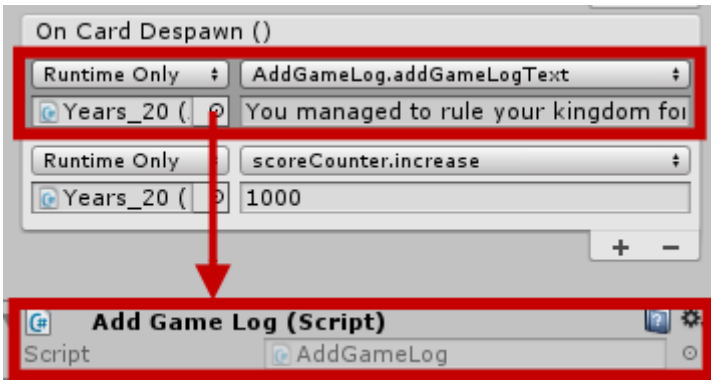
Game Logger (in GameLogger)



The Game Logger consists of two parts, one is the Game Logger itself, the other one is the Value Dependent Game Logs. It allows you to add a text log at the end of the game, which summarizes highlights and actions of the players actions/decisions.

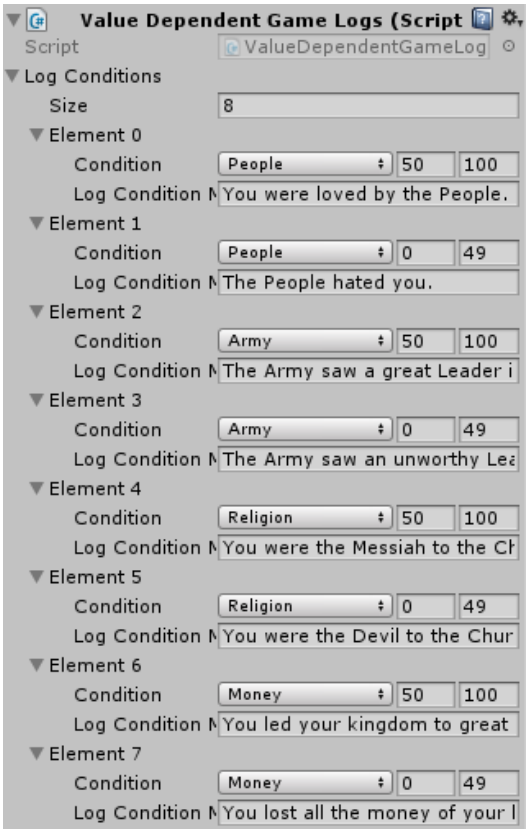
You can choose after how many text blocks a new paragraph should be created with **Text Break Every** for a clearer look.

To add a Game Log, simply add the AddGameLog script to the appropriate card, link it in the On Card Despawn Event of the EventScript and call the function: **AddGameLog.addGameLogText**
The text you enter in the below field will be added to the Game Log.



Value Dependent Game Logs

(in GameLogger)



The Value Dependent Game Logs are the other part of the Game Logs. They will be added after the normal Game Logs and are value dependent.

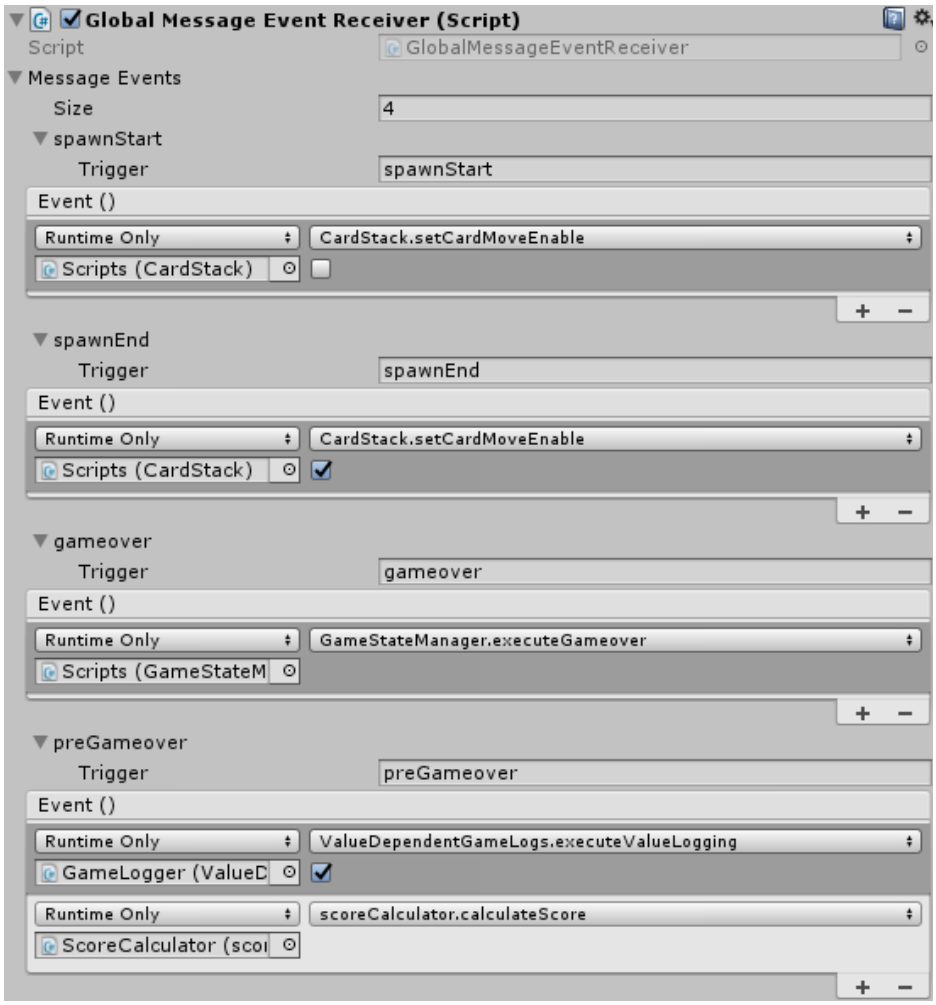
Which means you can select any value of your game and set a range, if the range is met at game over the appropriate text will be added.

In the example on the right side, there is an element for People 50-100 and another one for People 0-49. This means if the value of People is on Game Over above 50 the Game Logger will display the text: “You were loved by the People”. If it is below 50 it will display the text “The People hated you”.

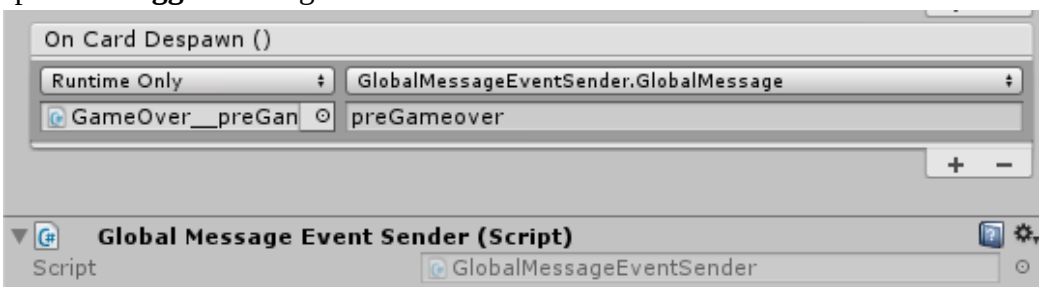
To Display the Game Log there is a ShowGamelog Script on the GameOver__Log Card.

Global Message Event System

The Global Message Event System consists of the 3 scripts: **GlobalMessageEventSender**, **GlobalMessageEventReceiver** and **GlobalMessageEventManager**. It is used to send Unity Events to spawned Prefabs without direct linking.

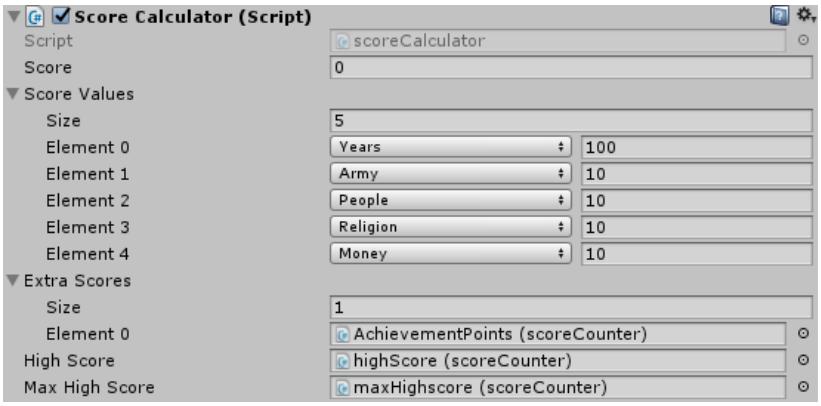


The **GlobalMessageEventReceiver** handles all send messages from every **GlobalMessageEventSender.** To send a GlobalMessage from a prefab (in our case a card) you simply need to add the **GlobalMessageEventSender** script to the card, link it in the **On Card Despawn Event** call the function: **GlobalMessageEventSender.GlobalMessage** and enter the specific **Trigger** message.



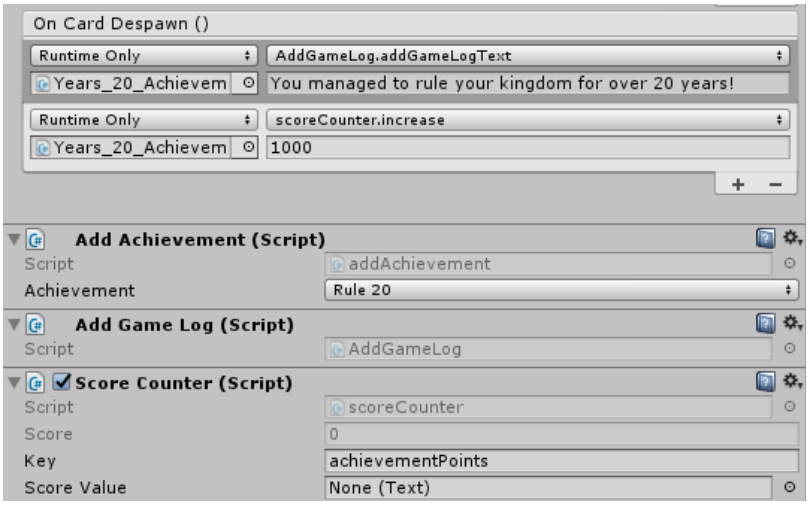
In this example the script is attached to the **GameOver__preGameover** card, when this card despawns the message **preGameover** is sent. The **GlobalMessageEventReceiver** will now run all events for preGameover, which is in this case executeValueLogging for the GameLogger and calculateScore for the ScoreCalculator.

Score Calculator



The Score Calculator calculates the player score at Game Over which is displayed on the GameOver__Score card. In the game example 5 values are used to generate the Highscore: Years, Army, People, Religion and Money. The number behind each value type is the score multiplier, this means the Years value has a multiplier of 100, e.g. years value is 12 then the score would be 1200 for Years.

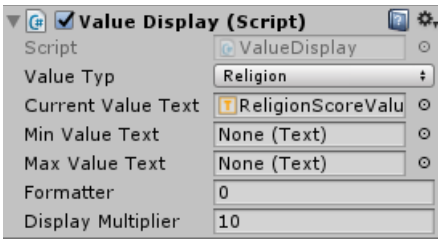
There is also the option to add **Extra Score** if you want to add points to the Highscore which are not value dependent. In this example they are Achievement Points. The achievement points need to be manually added from the cards (EventScript).



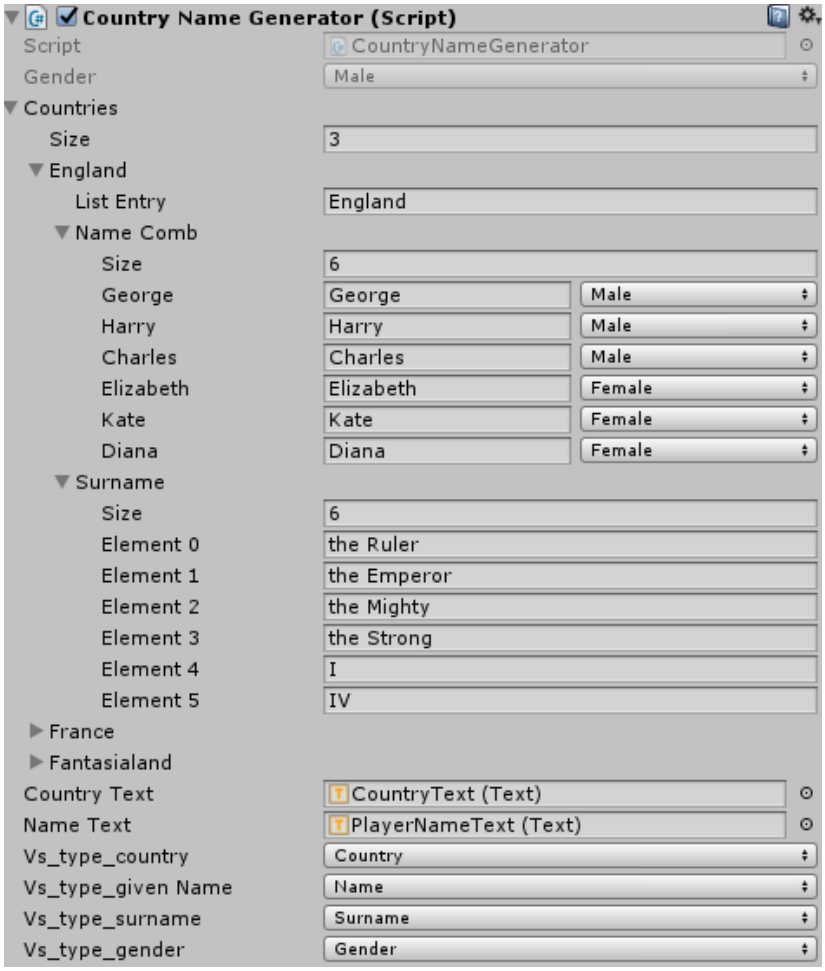
This example is from the **Years_20_Achievement** card. The Score Counter script has the key achievementPoints and is linked with the On Card Despawn event, which runs the function: **scoreCounter.increase 1000**, this adds 1000 points to the achievement points.

To display the achievement points or any other score, simply add a score counter script to the object you want it to display (in our example the GameOver__Score card) and link in the text field in Score Value and enter the appropriate key.

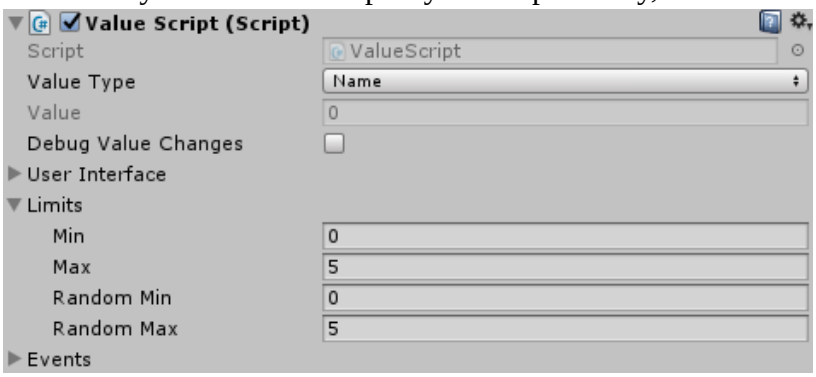
To display one of the base values you need to add the script **Value Display**. You can select the value you want to display from the drop-down menu, link in the text field where you want it to display. You can choose between the current value, and min/max value. Current value is for example used on the GameOver__Score card. The min and max values are displayed on the Highscore panel in the game menu, e.g. shortest reign / longest reign.



Country Name Generator (in Values)



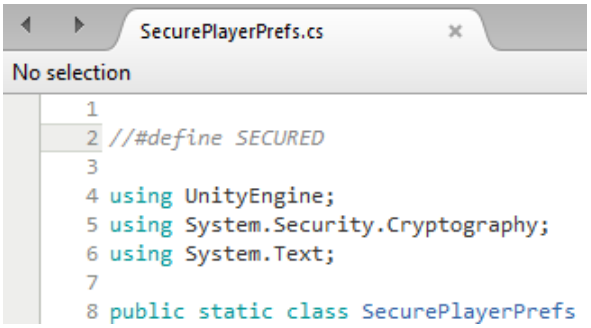
The Country Name Generator script is used to generate a combination of Name, Surname, Gender and Country. This should be pretty self explanatory, but there is one thing to note:



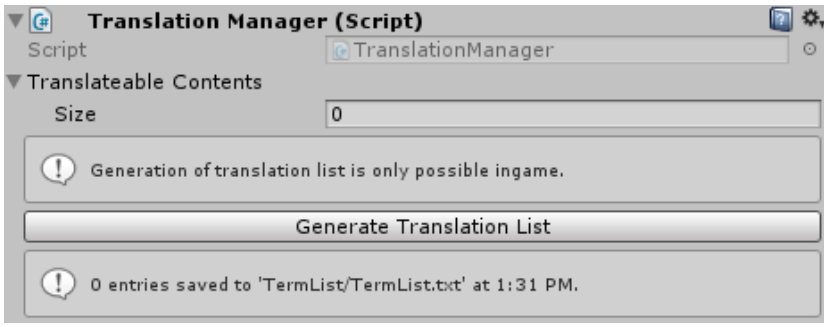
You should always use the same number of Names and Surnames for each country.
The reason for this is you have to enter in the value script of Name and Surname the Limits/Random min-max, in this example it is from 0-5, this means there are a total of 6 names per country (since it starts counting from 0). If you would now have a country with 8 names, the last two names would never be drawn. If you would set the range number for example from 0-10 and you only have 6 names it would draw the last name more often than the others because for every rolled value above 6 it would draw the last name in the list. But you can of course use 10 values for Name and only 5 for surname, because these values are separated from each other.

SecurePlayerPrefs (in scripts Project folder)

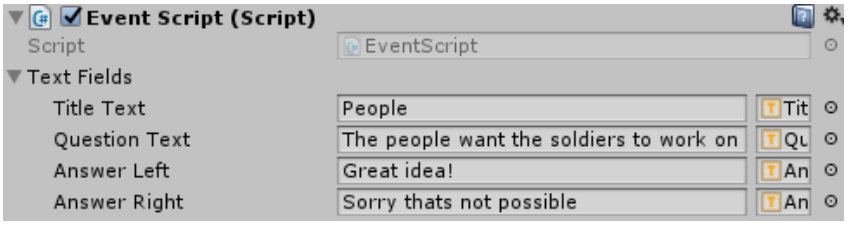
To prevent manipulation of the score or other values encryption is used. If you want to disable it you can simply open the **SecurePlayerPrefs** script and disable the second line, that it looks like this:



Translation Manager (in Scripts)



We have integrated support for the Localization Asset: **I2 Localization**
This means the text you enter in the EventScript of the cards will be used as terms and can be translated with I2 Localization.

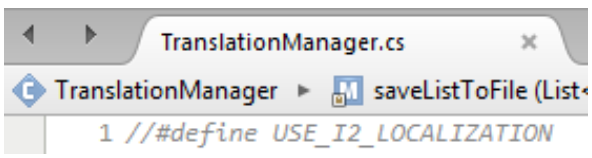


This also works for the Achievements, Country names and Game Logger entries.

With **Generate Translation List** a text file (Kings\TermList.txt) with all the text you have entered in these scripts is generated. You need to be in Play Mode for this. You can then open the text file in Excel (or any similar program) and copy it into the I2 Localization spreadsheet. This way you don't have to enter every term manually, but can import them all at once.

Please note: This does not work for the Game Logs added with the **Add Game Log** script. These have to be added manually to the I2 Localization terms.

To enable translation you need to import the I2 Localization Asset and enable the first line of code in the Translation Manager script.



Unity Ads (Update 1.10)

With Update 1.10 support for Unity Ads is now integrated. You can start playing it simply by calling the function **showAd** of the Play Unity Ad script.

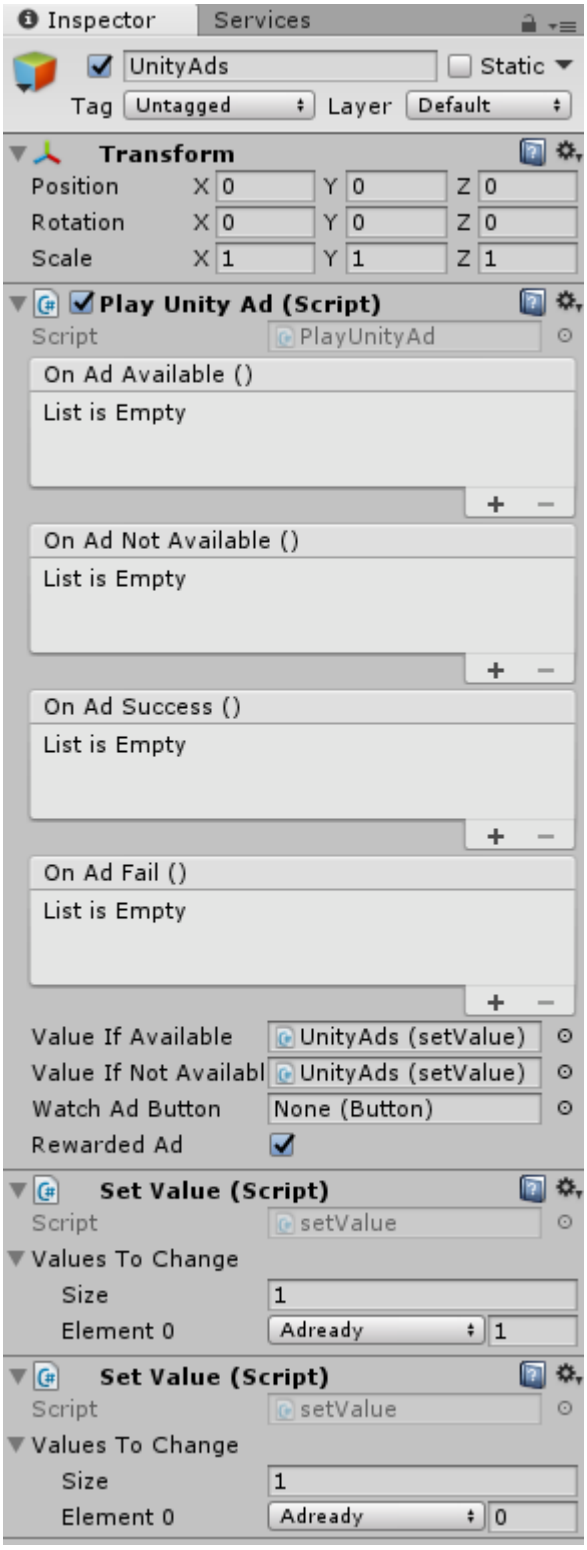
This script give you also Events for the following cases: Ad available, Ad not available, Ad success and Ad fail.

We also added a new value **Adready**. It is set automatically to 1 if an ad is available and to 0 if ads are not available. This allows the Card Stack to only draw Ad-Cards if an Ad is available.

If you uncheck **Rewarded Ad**, Ads can be canceled/skipped by the user.

In our example game two ad card have been created: **Ad_General** (appears one time randomly per game) and **Ad_People** (appears once when People value is 40 or below).

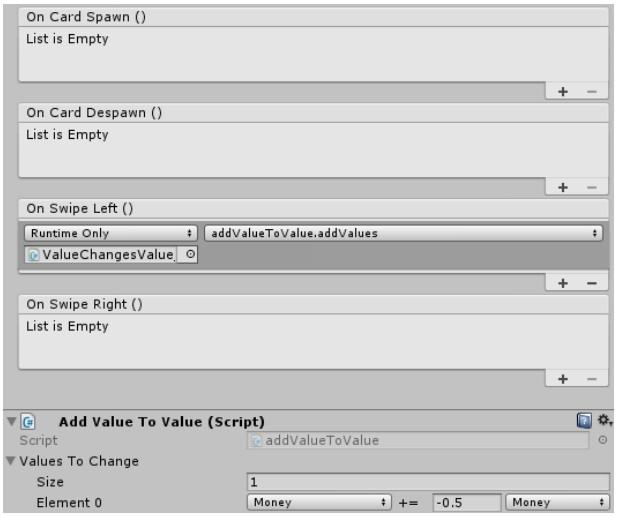
To enable Unity Ads go to the Services Tab (Ctrl+0) and turn Ads ON.



Add Value to Value (Update 1.20)

You can now directly add/subtract values from each other, with the **Add Value to Value** script.

You can see an example in the **ValueChangesValue** card prefab. In this example, half of the Money value will be deducted from the Money value (on swipe left)



Change Value over time (Update 1.20)

With the **Value Dependent Timer** script, you can now change values over time.

In this example the player will lose 1 health every 4 seconds, as long as he is married and his health is between 1 and 100.