



**UNREAL
ENGINE**



Paper 2D

Sprites & Class Blueprints

Steven Harris

TABLE OF CONTENTS

TABLE OF CONTENTS.....	2
1. Individual Sprites.....	3
1.1 Extracting Individual Sprites From a large Sprite Sheet	3
1.2 Sprite Collision Detection	4
2. Class Blueprints	6
2.1 Adding a Sprite to the Class Blueprint	7
2.2 Accessing a Variable in another Blueprint.....	8

1. INDIVIDUAL SPRITES

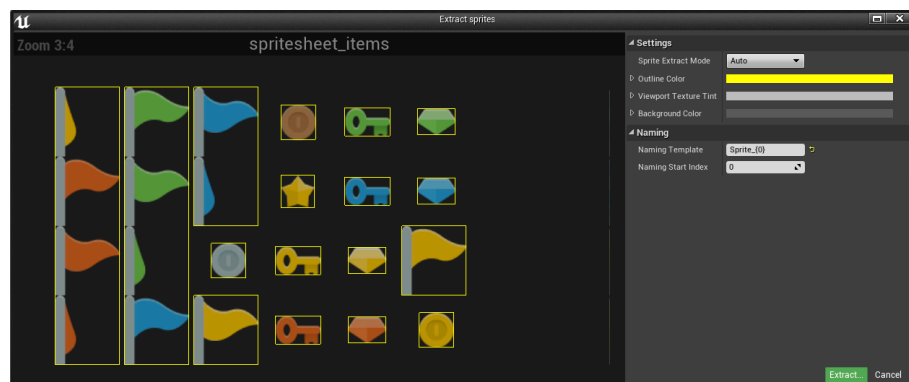
1.1 EXTRACTING INDIVIDUAL SPRITES FROM A LARGE SPRITE SHEET

It is possible to extract individual sprites from a large sprite sheet. To do this ensure that the sprites in the source sprite sheet are all organised in a uniform grid in the original image file. In the example below the tile grid is 128x128 pixels.

If you are extracting lots of sprites from one sprite sheet, you may want to place each sprite sheet into its own folder, to make organisation of the numerous sprites easier, as you will see below.

In the CONTENT BROWSER, select IMPORT, then navigate to the tile sheet and click OPEN. This imports the source image as a TEXTURE file into UE4.

Right click on the new texture and select SPRITE ACTIONS then EXTRACT SPRITES. The SPRITE EXTRACT MODE has two options: AUTO and GRID. Automatic extraction can work well if there is at least one blank row of pixels between the sprites. In the image below you can see that the smaller items have a yellow border all around them individually. This yellow border shows where UE4 will cut the sprites out. For the flags, the automatic detection has failed as there is no separation between the top of one flag and the bottom of the next. The detection algorithm therefore identifies all four vertical flags as one sprite. If your sprite sheet is constructed with at least one blank row and one blank column all around its edge, the automatic detection will work well.



To resolve this issue, you can select the GRID option in SPRITE EXTRACT MODE, as below. You will need to enter the individual sprite dimensions, in this case 128x128 pixels, which will create a cut out grid over the image and extract individual sprites from within each cell, as below.



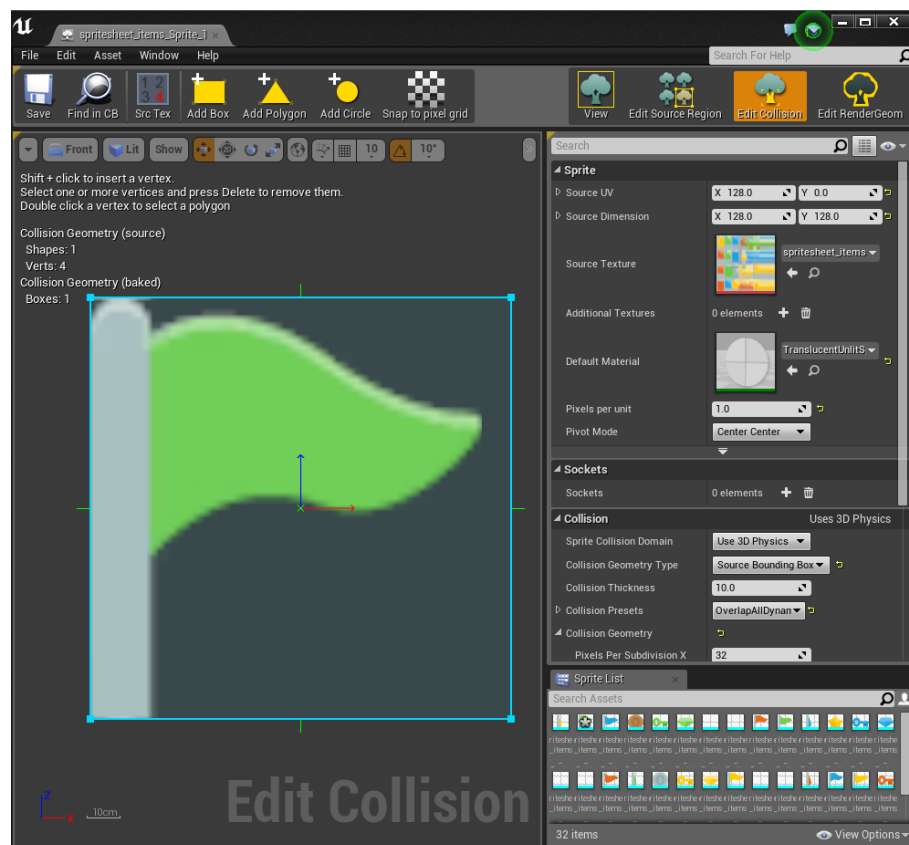
When ready, click the EXTRACT button. This cuts out the sprites and adds them into the same folder as the original sprite sheet in the CONTENT BROWSER. This will include any empty sprites, so you may wish to delete those.



The individual sprites can now be dragged onto the map as separate sprites. Make sure you change the Y axis value so that the sprites are at the correct depth in the scene.

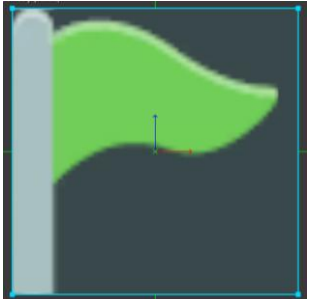
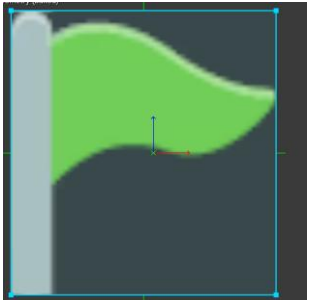
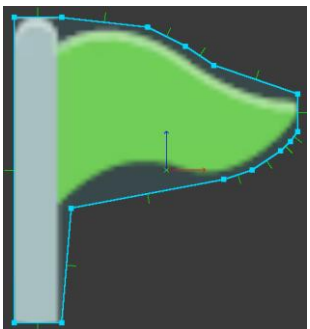
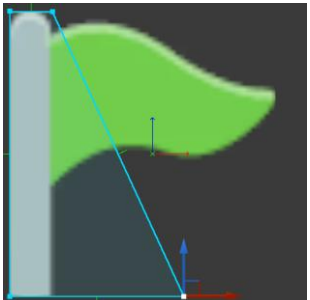
1.2 SPRITE COLLISION DETECTION

Double click a sprite in the CONTENT BROWSER to open the SPRITE EDITOR window, as below.



It may be necessary to edit the collision boundary of the sprite, as by default, collision detection will work on the entire bounding box of the sprite, even if the sprite is smaller, as in the example above.

In the COLLISION GEOMETRY TYPE option, there are several options which allow you to edit the collision geometry.

Option	Description	Example
Source Bounding Box	Collision box is set to the full dimension of the original sprite tile.	
Tight Bounding Box	The box will shrink to the extremities of the coloured sprite.	
Shrink Wrapped	UE4 will do its best to match the collision box to the shape of the sprite.	
Fully Custom	You have complete control over the shape of the collision box.	

In any of these modes you can click on a vertex and move it with the move gizmo to reshape the collision geometry. In doing this, the COLLISION GEOMETRY TYPE will change to FULLY CUSTOM.

The COLLISION THICKNESS option will determine how far into the next layers collision will work.

NOTE

When detecting collisions with sprites in Blueprint, you should use the **OnActorHit** event , and not **OnActorBeginOverlap**

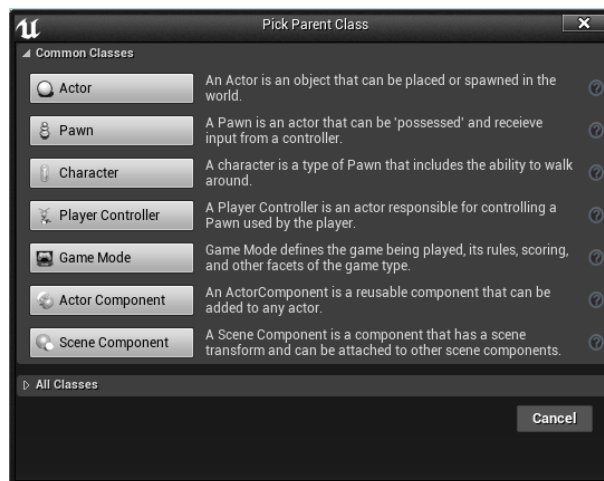
2. CLASS BLUEPRINTS

It is possible to add functionality into the Level Blueprint to control various Actors (game Objects). However, this is not efficient and would require the same code to be repeated several times for different instances of the same actor. Class Blueprints apply the same principles as traditional object orientated programming, in that a class can have its own variables, functions, events and in the case of UE4, its own 3D or 2D models.

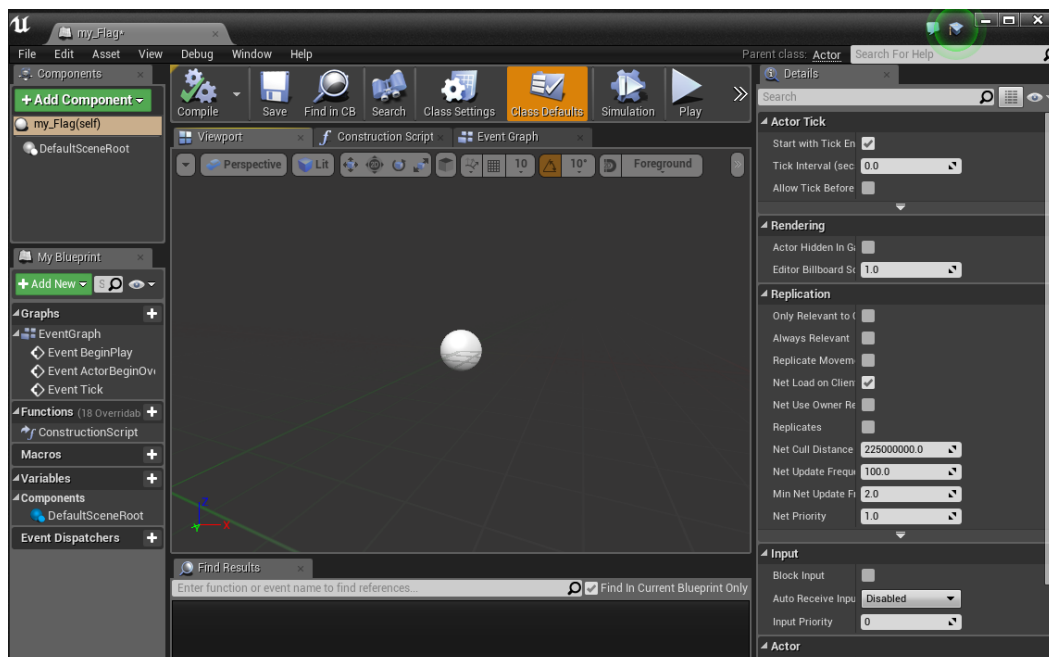
Rather than adding in 2D sprites and then creating code in the level blueprint, we should create a class blueprint for the actor, and add the sprite and blueprint code to that class blueprint.

Follow these steps to achieve this:

1. Select the BLUEPRINTS menus then select NEW EMPTY BLUEPRINT CLASS. This opens the PICK PARENT CLASS window



2. This window allows you to create a new blueprint based on an existing class, so that you can 'inherit' all the functions/variables etc of that parent class. This is useful if you need to create a new type of player character or player controller, and will save you some time in creating the basic functionality. In this case we will create a basic actor, so click the ACTOR button to base your blueprint on that class.
3. You will now be asked to select which folder in the CONTENT BROWSER you would like to save your new blueprint. It is advisable that you create a dedicated folder for your class blueprints. The Class Blueprint window will appear, as below



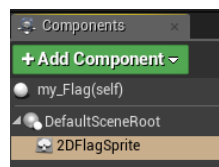
4. The various tabs are described below.

Components	Lists all the models, cameras and any other objects which exist in the game world.
My Blueprint	Lists all the functions, variables, macros, etc which belong to this blueprint.
Details	Displays any properties of the currently selected object in the Viewport.
Viewport	Displays any 3D models, cameras or any other assets which give this actor a physical presence in the game.
Construction Script	Blueprint code in this tab will only run when the object is placed in the editor. Often used to detect where an actor is placed in the level, and change its appearance accordingly.
Event Graph	This is where the core blueprint is placed.

2.1 ADDING A SPRITE TO THE CLASS BLUEPRINT

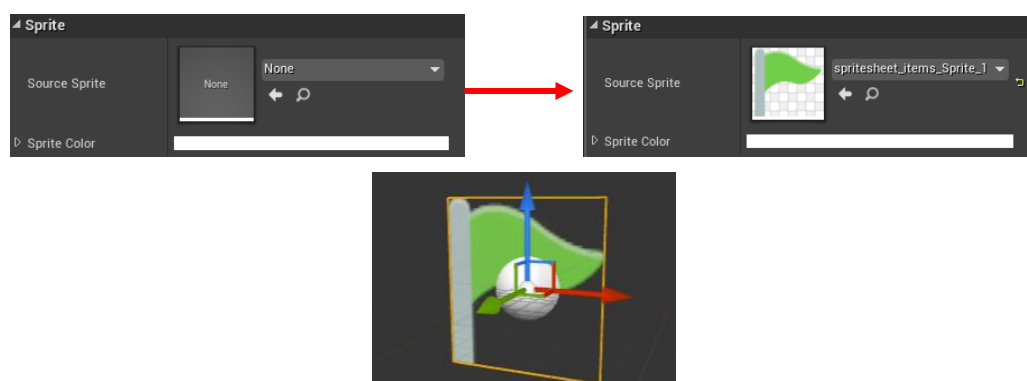
To give this Actor a physical presence in game, we need to add a sprite. Before proceeding, make sure your sprite is setup as described earlier.

In the COMPONENTS tab of the class blueprint, select the green ADD COMPONENT button. Then select the PAPER SPRITE option and give the sprite a name, in the example below, this is called 2DFlagSprite.



This has created an empty paper sprite entry. We need to link this to a sprite we have imported, as described earlier. In the CONTENT BROWSER, select this sprite you want to add to the actor.

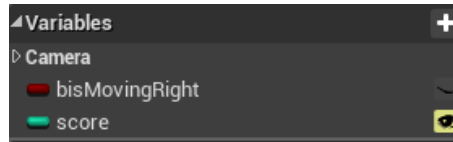
In the DETAILS tab, go to the SPRITE section and click the white arrow next to the SOURCE SPRITE option to apply the selected sprite. This will then appear in the VIEWPORT tab.



With the sprite selected, you can now move to the EVENT GRAPH tab and add blueprint code as normal.

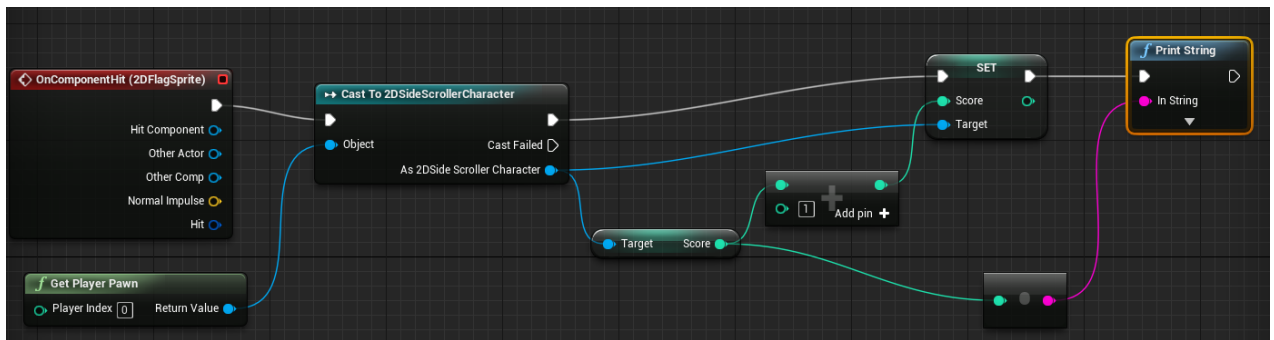
2.2 ACCESSING A VARIABLE IN ANOTHER BLUEPRINT

It is often necessary to access a variable in one Blueprint from another Blueprint. For example, the player score variable may be contained within the PLAYER BLUEPRINT, but the PICKUP blueprint needs to change the score each time a pickup is touched by the player. In this example, the score variable would be created in the player blueprint and made visible to other blueprints by clicking the eye icon next to the variable name so that a yellow eye is visible, as below.



Take a note of the name of the blueprint in which this score variable is stored, as you will need it in the next step.

In the blueprint which needs to access this variable, right click on the blueprint and in the command list, type without quotes: “cast to<name of blueprint with score>” for example “cast to 2DSideScrollerCharacter”. Then select the node which matches that text, as below. Now add a GET PLAYER PAWN node and connect this to the CAST TO node as below. The player pawn is the object that the player is currently controlling.



The CAST TO node is effectively checking that the Pawn returned from GET PLAYER PAWN is a 2D SIDE SCROLLER CHARACTER. In our game this will always be true, but in some games the player pawn can change from a character to a vehicle, so we need to check before we can access any information in that blueprint.

Once you have a CAST TO node for your appropriate blueprint you want to access, you can then click and drag from the AS 2D SIDE SCROLLER CHARACTER pin, and search for the variable you need to access. You will be presented with a GET or SET action when you select the variable.