INSTITUTE OF TECHNOLOGY
BLANCHARDSTOWN

A Taster of Computing
[[VERSION – Unity 2D – C# language]]

Gravity Guy 2D (2014) - a little computer game...

Part 4 – background image by popular request …

1 Aims of this part of the tutorial

1.1 New features / skills to be learned in this part of the tutorial

In this part of the tutorial you will add the following features to our game:

- *Background image (move with camera?)*

2 Add background and move behind contents of scene

2.1 Add background image sprite to scene

First drag a copy of the background image sprite into the scene

- Drag image from Project ‘Sprites’ folder into the scene

2.2 Set z-position BEHIND other scene gameObjects

The 'Z' axis relates to how near / far an object is to the user. Negative values are nearer the user, positive values are further away. The defaults are:

- Camera has z = -10
  - So camera can 'see' any object with a value from -9 to 0 and to any positive z-value
- Other game objects have z = 0
  - So they can be 'seen' by the camera

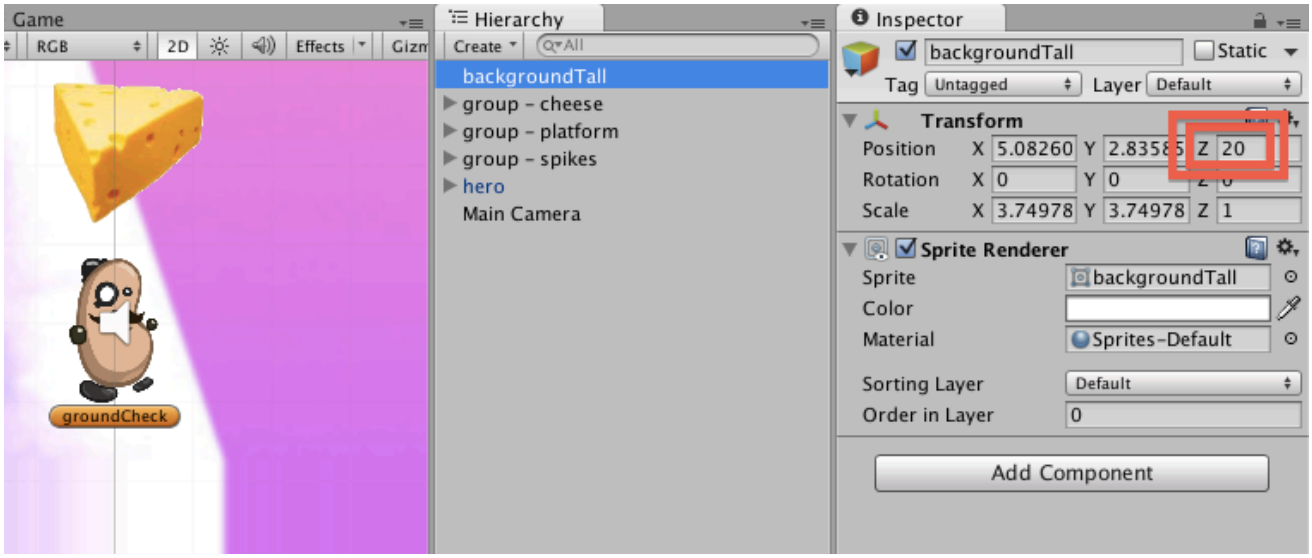
NOTE – because we are in 2D mode we have an 'orthographic' camera

- This means that objects are 'projected' onto the camera the same size, regardless of how far away they are
- In 3D we have a 'perspective' camera
  - Which makes far away objects smaller, and nearer objects larger …

Set the z-vaue of the background image to 1

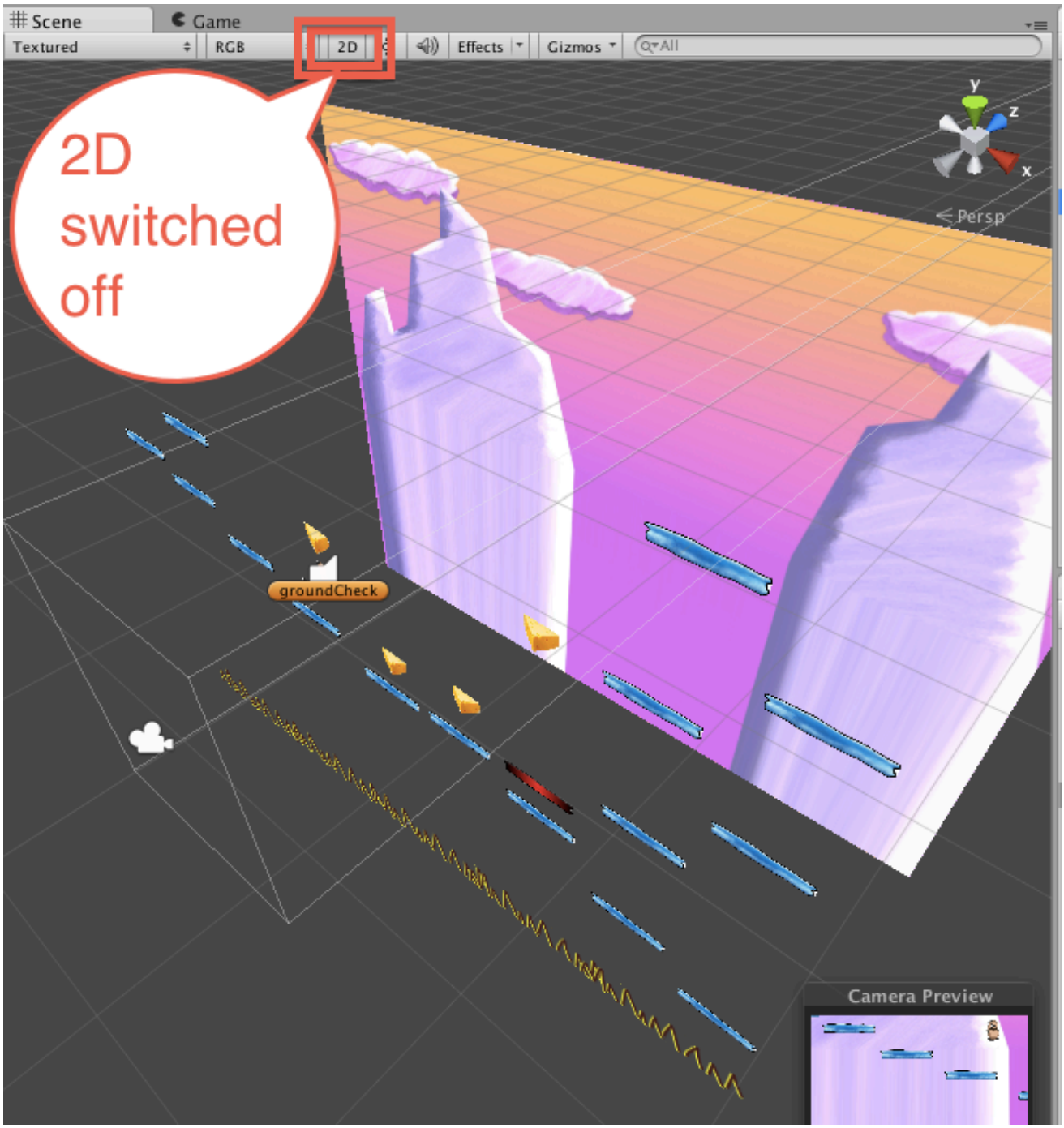
- Select background image gameObject in **Hierarchy**
- In **Insepect** set the z-value of its transform.position value to 20

You should now see all your cheese, platforms, spikes, hero guy IN FRONT of the background image:



2.3 EXTRA EXTRA – switch TO 3D view to SEE objects with z-values

If you turn off '2D' mode in the **Scene panel**, you can see how the objects are positioned by the x,y and z values:



NOTE – controlling what you see in the Scene panel in 3D mode takes a bit of practice, try this:

- Select (double click) the 'Main Camera'
- Use ALT-mouse-drag to rotate what is viewed
  - I find having camera on the lower left, 'looking' toward the upper right (positive Z) a good arrangement
- You may need to ZOOM-out to see most objects
  - Either use the mouse scroll wheel to zoom
  - OR you can go into HAND-tool mode (press Q), and then use CTRL-mouse-drag to zoom in/out

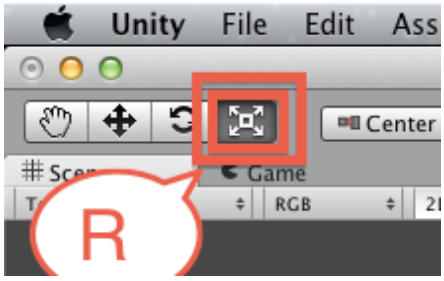
3 Size background so it is behind all platforms / scene contents

3.1 Resize your background image

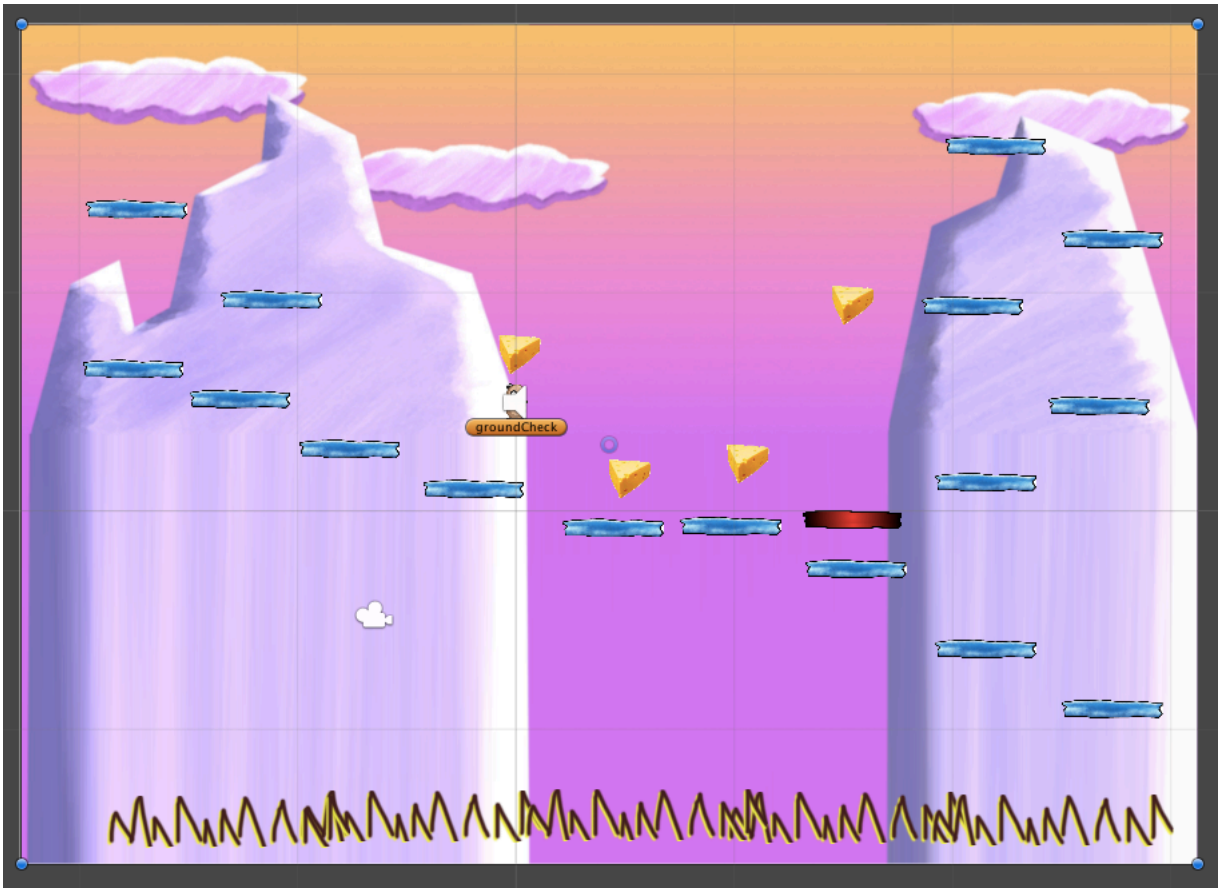
You may need to 'stretch' the image, so that when the game is player no 'blue' camera default background can be seen.

Do the following:

- Populate your scene with platforms / pickups / death traps as desired
- Position and resize the background image
    - METHOD 1: by hand
        - Ensure the background gameObject selected, and its 'Sprite Renderer' components properties are displayed (the little triangle in the **Inspector** for this component should be pointing DOWN) – this ensures you have the move circle and drag-handles active
        - In HAND-tool mode (press Q) select the background and move it around with its center circle, and resize with the blue drag-handles in the four corners
    - METHOD 2: use the scale tool
        - First set the transform-position of the background image to (0,0, 20)
            - It's always a good idea to have (0,0,0) as the centre of your scene
        - Second, go into SCALE-tool mode (press R) or click the icon with the square and 4 corner arrows coming out of it



        - Ensure the background gameObject is selected
            - You should see a WHITE square in its centre
        - Scale the background image by clicking and dragging from the centre of this white square
    - THE ADVANTAGE of METHOD 2 is that you preserver the POSITION of the object at (0,0,20)
        - Method 1, because you are resizsing by a corner, means the centre position changes as the gameObject is resized
            - Meaning you have to reset it back to (0,0,20) after each resize if you want to keep your scene nice and tidy

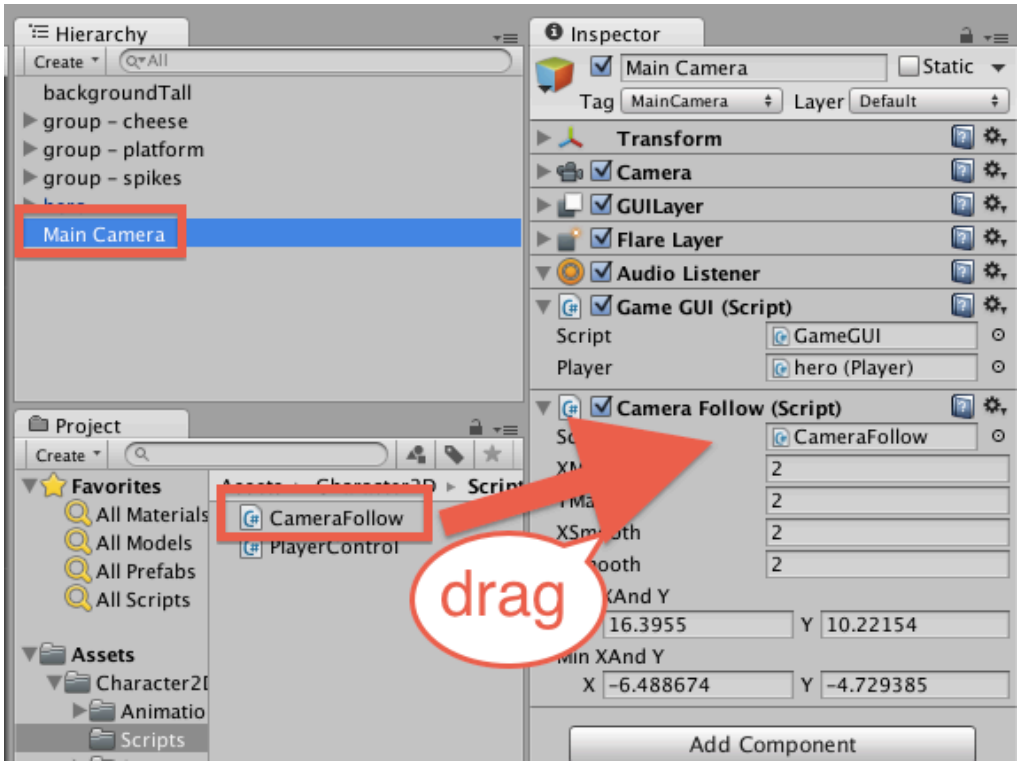

groundCheck

4 Add ‘CameraFollow’ script to the camera, and set X/Y limits

4.1 Add the basic CameraFollow script to the Main Camera

Let’s add a pre-written script to the camera, so that it moves when the player’s ’hero’ travels to one of the edges of the screen:

- In **Hierarchy** select Main Camera
- Drag ‘CameraFollow’ script from **Project panel** folder Assets – Character2D – Scripts into the **Inspector**
  - Or directly onto the Main Camera in the **Hiearchy**
  - Both methods add an instance of the script class as a component of the Main Camera GameObject



4.2 Playtest your game

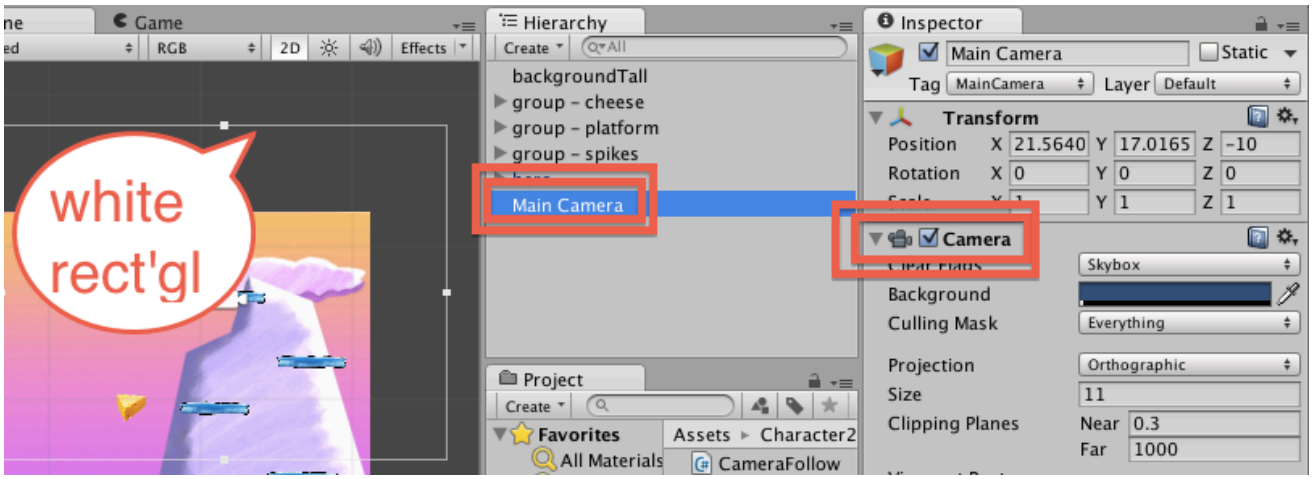
You should find the camera now moves to follow the camera, when the player moves some distance away from the center of the window.

BUT: it is likely that the camera limits are not correct – i.e. the script has a MAXIMUM and MINIUM X and Y setting, beyond which the camera will never be allowed to move …

4.3 Calculate your scene's MAX and MIN X and Y values

Here is my personal procedure to calculate the 4 values (maxX, maxY) (minX, minY) that will ensure the camera will allow your player to see all around your scene, but never beyond the scene contents. Do the following:

- Select the Main Camera in the **Hierarchy**
  - And ensure its 'Camera' component has its properties displayed in the **Inspector**
    - This ensures you can see the WHITE RECTANGLE showing the part of the scene the camera can 'see' and will be displayed in the **Game panel** when the game runs

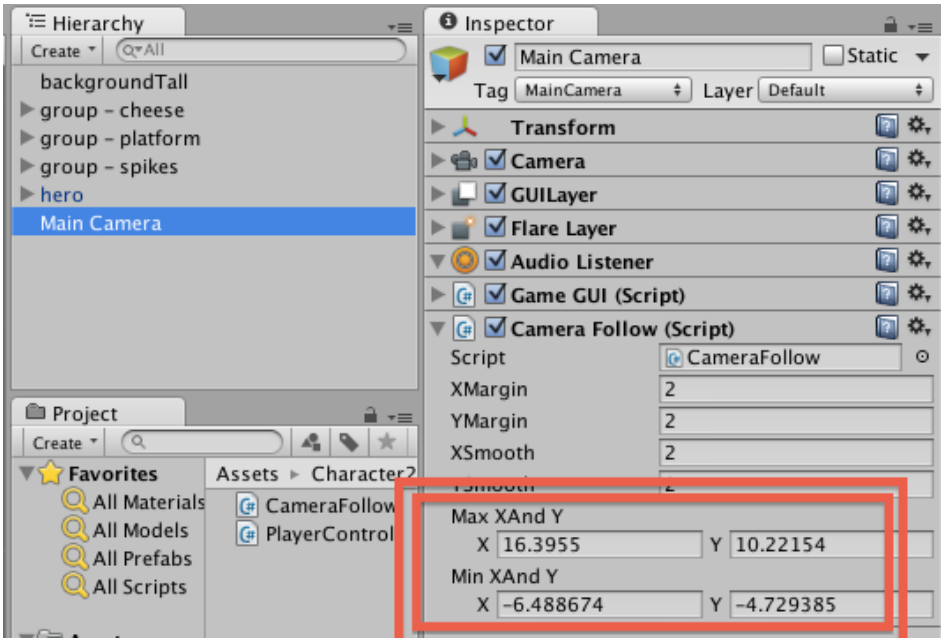


- Select the MOVE-tool (press W)
  - You should see the X and Y move arrows on the camera in the **Scene panel**
    - The colours of the move arrows can be remembered as follows:
      - X, Y, Z
      - R, G, B (from RGB colours – red, green, blue)
      - So X is RED and Y is GREEN
- Find the MAXIMUM X- and Y- values for our camera movement
  - Move the camera so that its TOP RIGHT white rectangle is at the TOP RIGHT of your background image
  - Make a note of the X and Y values of the camera's transform-position
    - For me it was: X = 16.3955, Y = 10.22154
    - TIP: copy and paste the exact decimal values from the **Inspect** into a blank text editor window
- Find the MINIMUM X- and Y- values for our camera movement
  - Move the camera so that its BOTTOM LEFT white rectangle is at the BOTTOM LEFT of your background image
  - Make a note of the X and Y values of the camera's transform-position
    - For me it was: X = -6.488674, Y = -4.729385

4.4 Set your scene's camera limits in the Inspector

Now we know the numbers to type in for the properties of our CameraFollow script component:

- Select the Main Camera in the **Hierarchy**
- In the **Inspector** for the Camera Follow component, type in (or copy/paste) the maximum and mimum X and Y values you noted from the previous step



Now when you run the game, you should find the camera can follow the hero guy all around your scene, but never moves too far left/right or up/down, using the limits we have entered

NOTE – you may have to add a few extra platforms / move the spikes, to fully test camera movement in your scene