1) **Importing the landscape**

a) Each pixel represents a square meter, so the X and Y needs to be scaled appropriately to match.

i) Each pixel in the DEM represents ***7.1854***x***10.2917*** meters, so the landscape will need to be scaled by setting X to 718.54 and Y to 1029.17 (both starting from 100).

- This means that some sort of processing power will need to be applied to the image, as scaling it up means that all the new pixels in between the original (now spread out) pixels need to be dealt with. We found the cubicspline method (from the GDAL library in python) to be the best for our purposes.

ii) UE's maximum resolution for a heightmap is **8161**x**8161**, so the image file will need to be smaller than this. Given that the original DEM (when scaled appropriately) becomes **11597**x**19904** pixels, it will need to be reduced. The smaller the resolution the better performance UE will have, but this means the water used in the water plugin becomes more pixelated for some reason. (Is this a bug?)

- Maximum scale I can do (worst performance) is scaling the DEM down by 2.45 (making the image **4733**x**8124**), then scaling it in UE by setting both X and Y to **245**.
- Original scale I can used, which didn't slow down UE too much and wasn't too pixelated with water, is scaling the DEM down by 4 (making the image **2899**x**4976**), then scaling it in UE by setting both X and Y to **400**.
- Least scale I can do (best performance, but most pixelated with water) without losing any detail in the DEM is scaling it down by 7 (making the image **1656**x**2843**), then scaling it in UE by setting both X and Y to **700**.

b) When a grayscale image is used as a heightmap, UE maps its minimum and maximum heights to -25500 and 25000 cm, respectively. As such, if this does not represent the minimum and maximum heights of the heightmap, it will need to be appropriately scaled along the Z axis. This is done by obtaining the ratio between the height range of the DEM to that of UE, shown by the following equation:

$$\frac{DEM\_MAX - DEM\_MIN}{UE\_MAX - UE\_MIN} = RATIO$$

i) The minimum and maximum heights of the DEM is **-614.33** and **3801.78** meters (or -**6143.3** and **38017.8** centimeters), respectively. To have UE accurately show the right scale, we will need to get the ratio of the DEM range (**-6143.3** to **38017.8**) to the UE range (**-25500** to **25500**), which is shown in the following equation:

$$\frac{38017.8 - -6143.3}{25500 - -25500} = .865903922$$

As such, the scale value for the Z axis in UE needs to be set to **86.5903922**.

c) Lastly, the Z position of the heightmap will need to be adjusted so that the sea level (height of zero) in the DEM is set at sea level in UE (as the previous step only affects the heightmap Z scale and not where it is actually positioned). This is obtained by performing 2 simple operations: moving the landscape up by half of the DEM range (to get the bottom of the heightmap at sea level), then moving it in the direction of the minimum height, shown by the following equation:

$$\frac{DEM\_MAX - DEM\_MIN}{2} + DEM\_MIN = LANDSCAPE\_HEIGHT\_IN\_UE$$

i) Currently, the bottom of the heightmap is at a height of **-22080.55** (which is half of the range between **-6143.3** to **38017.8**), when it needs to be at **-6143.3**. As such, we need to move the landscape up by **22080.55**, then down by **6143.3**, which is shown in the following equation:

$$\frac{38017.8 - -6143.3}{2} + 6143.3 = 15937.25$$

As such, the Z position for the landscape in UE needs to be set to **15937.25**.

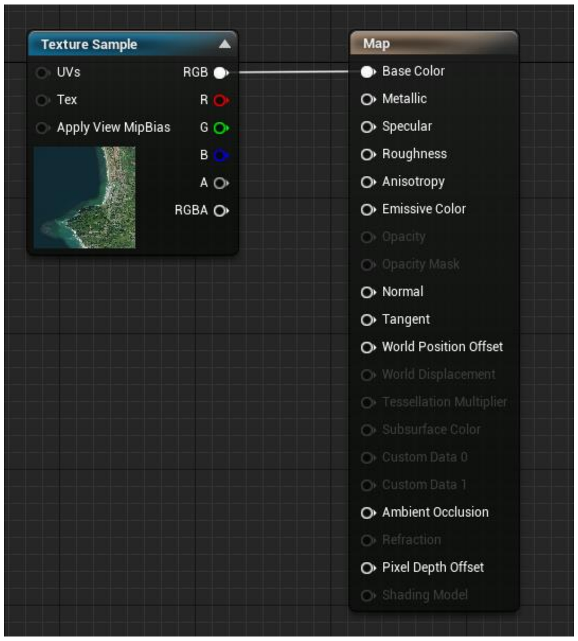
d) In addition, when intersecting the landscape with a plane at sea level, I personally noticed that the landscape seems to match the original DEM's shape most accurately when I move the plane down by **62** centimeters. Interestingly, if I moved the landscape *up* by **62** instead, the total Z position becomes **15999.25**, which is really close to **16000**. Is this a coincidence? Given that the 64-bit DEM was converted to a 16-bit image, perhaps there was a loss of precision? Is there something special about **16000**?

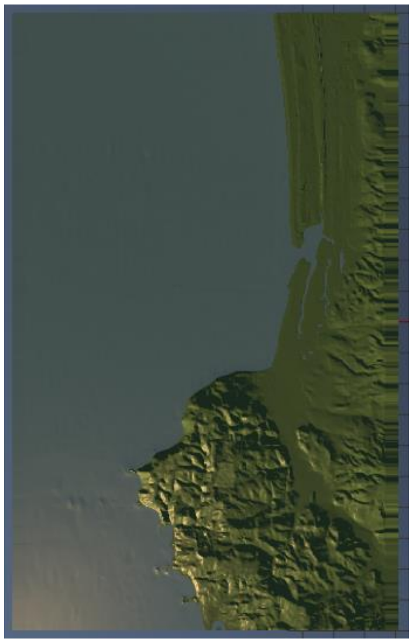
2) Texturing the landscape

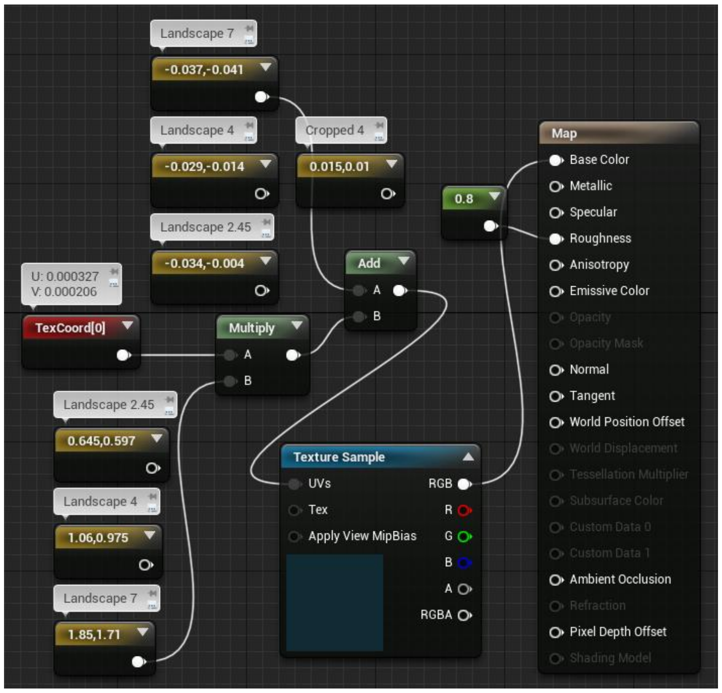
a) Zach obtained the satellite image for our landscape so I can't explain that process, and so the next step is to apply that texture to our landscape. However, it isn't as simple as simply dragging the image onto it:

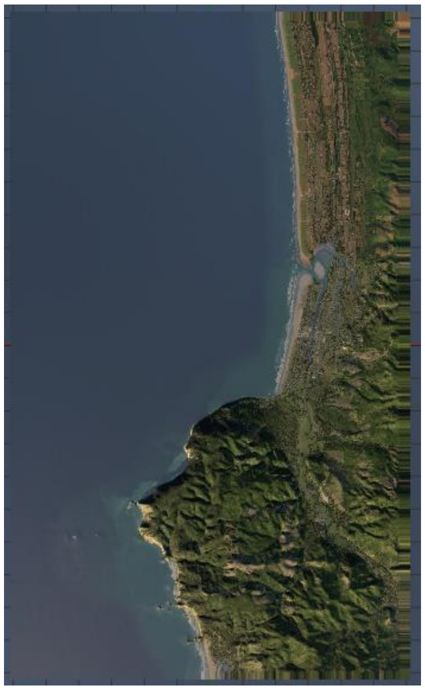
i) First, we created a new material and inserted a TextureSample node (which takes an image as input) and linked its RGB output to the Base Color input of the Result node (first image below). After saving and applying the material to the landscape, however, it does not provide the intended look (second image below).

ii) After much experimentation, I realized that this was caused by some sort of issue regarding how the texture coordinates were mapped to the landscape, and it was different for each heightmap I used (the images scaled by 2.45, 4, or 7). The values I used for each are shown in the third image below, with the result on the landscape in the final image (note that I also affected the texture's Roughness so that it does not reflect as much light). The general material was also created by Zach, so I don't know his process for organizing the nodes, nor why the TexCoord node's values are what they are: I just modified the multiplication and addition values.









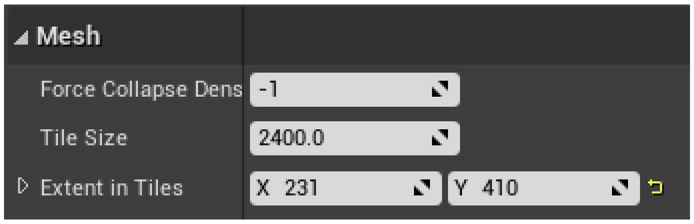
3) **Adding Water**

a) With the new 4.26 update, UE added a new water plugin that we wished to use for this project. The border for each type of water body is controlled by splines, which needs to be manually drawn out by the user (as far as my knowledge - I don't know how to automate this process). Anytime I would spend a long time adjusting or creating splines, I would hide the Water layer in Landscape mode in order to increase performance. Unchecking "Affect Landscape" in the WaterBrushManager works as well (I discovered this later).

i) The first image below showed what happened when I traced [most of] the border of the Pacific Ocean. It did modify the landscape at the curves, but the water mesh would only stay within the highlighted square below and did not extend to the rest of the terrain – and scaling the X and Y axes did not seem to change anything. To solve this issue, I needed to learn how the WaterMeshActor worked, as I quickly discovered that this actor is completely separate from the ocean actor (WaterBodyOcean) and entirely controls where the water is. After a lot of experimentation and searching, I discovered I could either change the "Tile Size" or "Extent in Tiles" to extend its border to the rest of the terrain:

- Modifying "Tile Size" would make the tile itself bigger, thus only scaling the geometry (as opposed to adding more vertices), but with the side effect of stretching the water's waves so that they become more spread out (thus risking losing the realism of the waves). Perhaps this could be fixed by modifying the parameters regarding the wave dynamics, but I haven't yet spent time learning about this and there quite a bit to learn, and this is not yet a main feature I'm looking to add – so testing this will be delayed until after I get everything working.

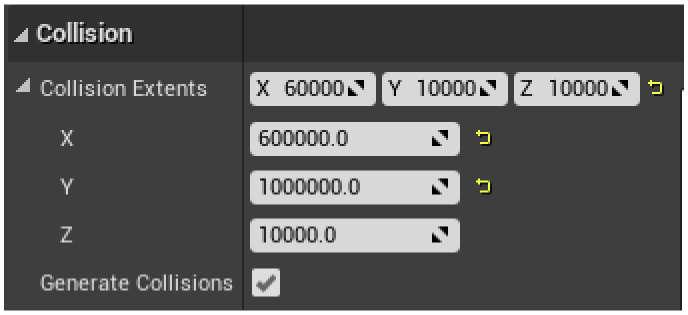
- Modifying "Extent in Tiles" controls how many tiles there are along the X and Y axes, thus changing the amount of geometry and waves to be calculated (which decreases performance). I ended up choosing this method (I could have used both if I wanted to, it doesn't have to be one or the other), as the amount I changed them by did not noticeably affect performance too much on my machine (both values were originally **64**):



Mesh

Force Collapse Dens	-1
Tile Size	2400.0
Extent in Tiles	X 231 Y 410

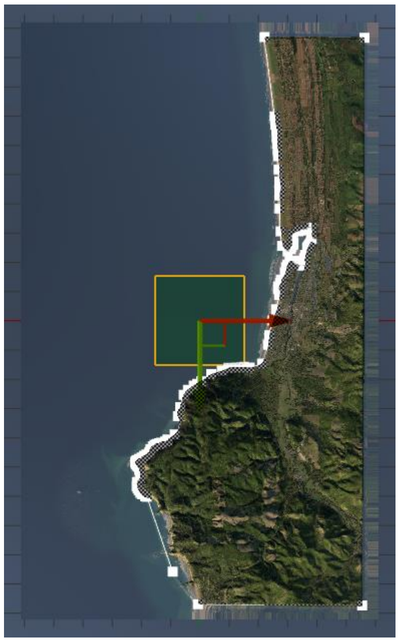
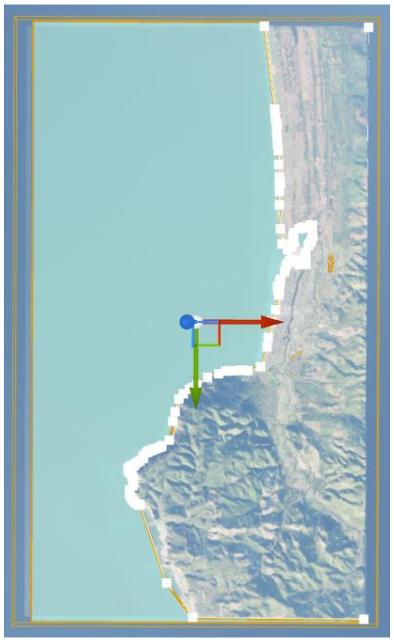
However, while the water was now where it needed to be, I noticed that it only rendered the underwater effect in the exact same zone as before, and moving out of that square would lose the effect. Some more experimentation later, and I learned that I needed to modify the WaterBodyOcean's "Collision Extents", as it only renders the post processing effects within them (both the X and Y values were originally **50,000** – I didn't change the Z value):

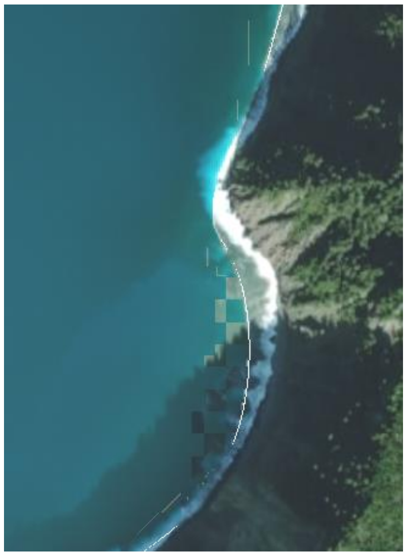
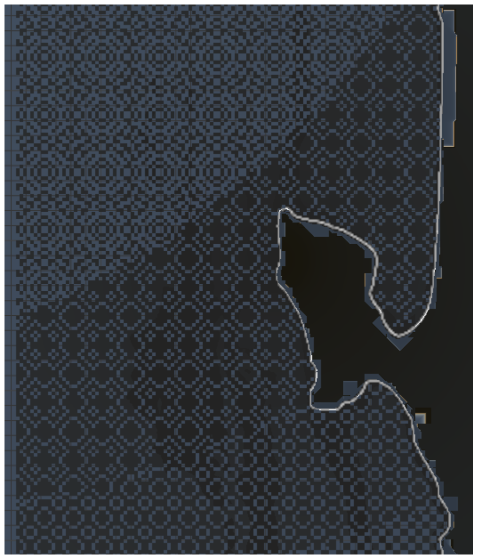
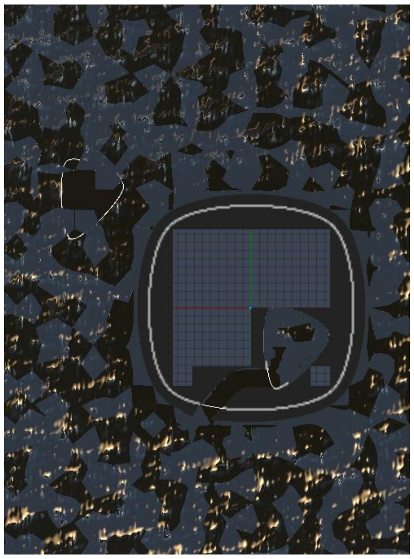


Collision

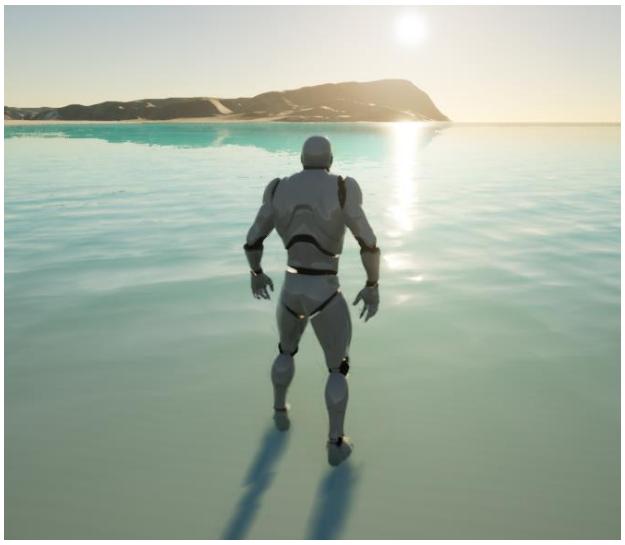
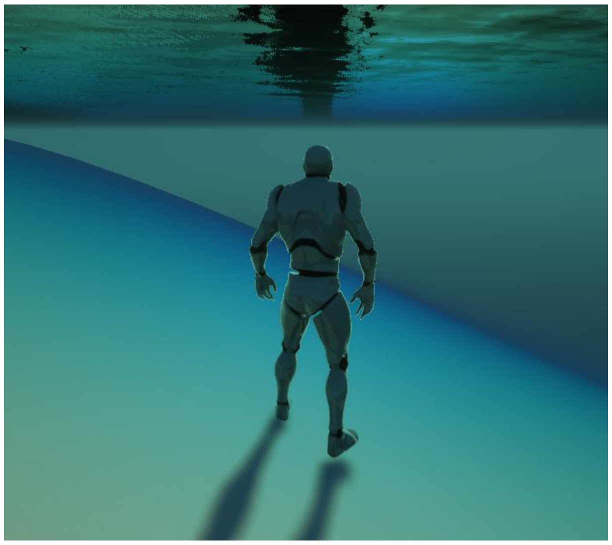
Collision Extents	X 60000 Y 10000 Z 10000
X	600000.0
Y	1000000.0
Z	10000.0
Generate Collisions	✓

The second and third images below show how the landscape was modified in this process (note the 2 different orange boundaries surrounding the terrain: one represents the WaterMeshActor's tile extents and the other represents the WaterBodyOcean's collision extents), which is definitely what was intended. However, I noticed that the water mesh seemed to break in certain areas (this becomes especially clear when viewing the entire landscape from the bottom), which I suspect is a bug in the plugin, as it doesn't show up when making a demo level using the default UE landscape (shown in the last 3 images, from left to right: area where the water mesh is breaking in an area where it shouldn't; view of landscape from the bottom [the dark part is the ocean and the circularly tiled part is the landscape); and view of demo landscape from bottom [the terrain is right in the middle with the light-reflecting ocean all around it]):

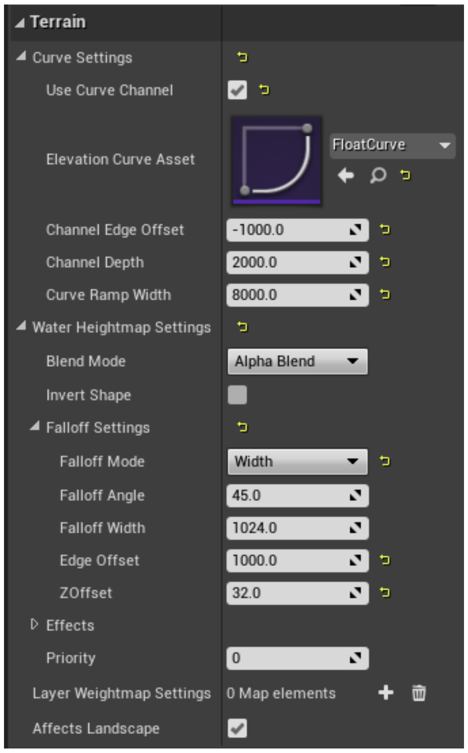



I also realized that the WaterBodyOcean would significantly deform the landscape underwater – to the point that it does not accurately represent the terrain, as shown in the comparison between the first two images below (both located at the same position and orientation: the first is what I want it to be [which was implemented by simply hiding the Water layer in Landscape mode] and the second is what it turned into when I added the WaterBodyOcean).

The WaterBodyOcean actor has a lot of parameters that could be adjusted, so I experimented with them to try to get what I was after (screenshot of original values shown on the right):

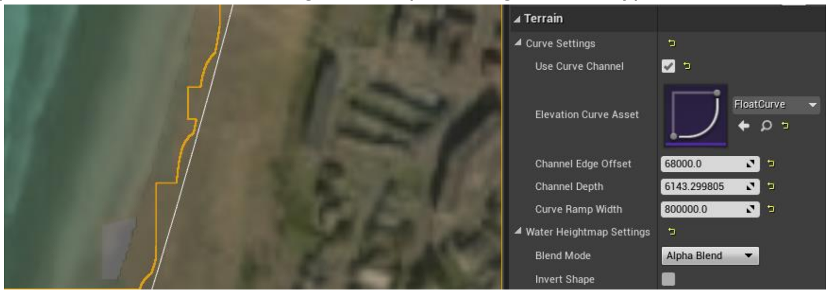


Terrain	
Curve Settings	
Use Curve Channel	✓
Elevation Curve Asset	FloatCurve
Channel Edge Offset	-1000.0
Channel Depth	2000.0
Curve Ramp Width	8000.0
Water Heightmap Settings	
Blend Mode	Alpha Blend
Invert Shape	☐
Falloff Settings	
Falloff Mode	Width
Falloff Angle	45.0
Falloff Width	1024.0
Edge Offset	1000.0
ZOffset	32.0
Effects	
Priority	0
Layer Weightmap Settings	0 Map elements
Affects Landscape	✓

- First thing I tried was unchecking "Affects Landscape", but this simply got rid of the water meshes (but it did return the landscape back to its original heights - is the lack of water a bug?)

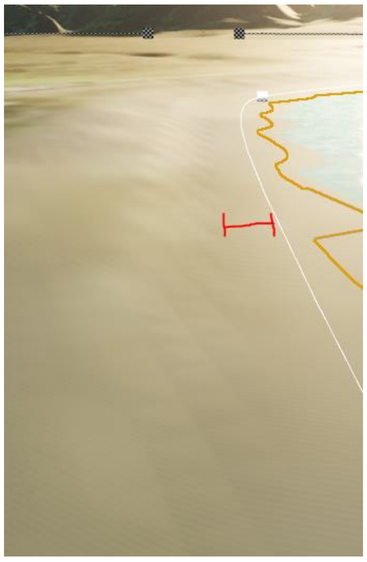
- Next thing I tried is unchecking "Use Curve Channel" (as this is directly responsible for the shape of that curve underwater), but this had the exact same effect as the previous attempt (is this a bug?)

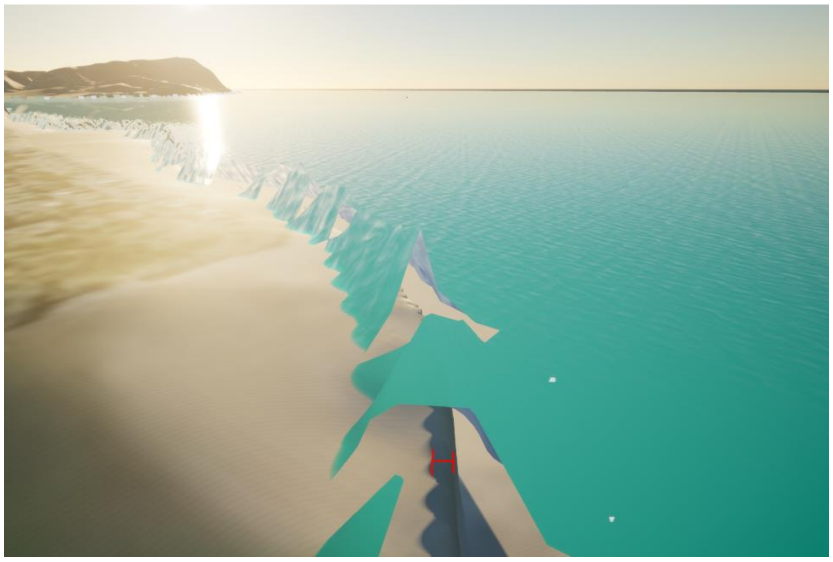
- Next, I tried modifying the Channel parameters to that it accurately represents the landscape, shown in the image below. This got close to the landscape effect I was after (the underwater terrain sloped downwards too slowly), but the main issue now became that border between the ocean and the terrain was visibly pixelated, as seen in the image below (the orange boundary):

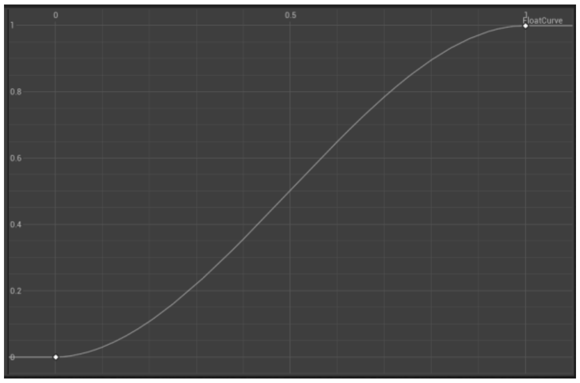


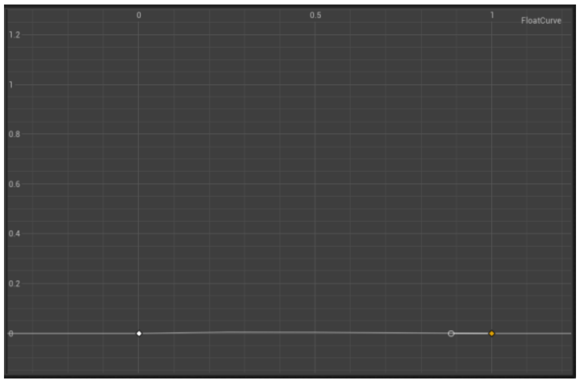
Terrain	
Curve Settings	
Use Curve Channel	✓
Elevation Curve Asset	FloatCurve
Channel Edge Offset	68000.0
Channel Depth	6143.299805
Curve Ramp Width	800000.0
Water Heightmap Settings	
Blend Mode	Alpha Blend
Invert Shape	☐

- Next, I tried modifying the Water Heightmap Settings in hopes of fixing the pixelation error (the falloff is hard to see, so I added an indication in the first image to identify what it was). The first thing I did was change "Blend Mode" to "Min" in hopes that it wouldn't modify the landscape underwater, but I didn't notice any effect I was after – and this was the same with "Max" in hopes of removing the falloff effect. In addition, I didn't notice any difference when checking "Invert Shape". However, changing it to "Additive" made a significant difference (where, instead of replacing the landscape, any deformations are added instead), but it didn't get the effect I wanted either: it just made the falloff *really* obvious, and the underwater terrain beyond that became completely flat (rather than showing the original slope of the terrain – shown in the second image). I also tried changing the Elevation Curve Asset so that, instead of an interpolated curve from 0 and 1, it was a flat line at 0 (shown in the third and fourth images), but this had the same effect as my first two attempts and the water mesh completely disappeared. In addition, either by changing "Blend Mode" to "Additive" or modifying the Elevation Curve Asset, it caused these mountainous water meshes on the border of the shoreline, which I could not get removed even after changing all the parameters back to their original values… I could reload the level without seeing that issue, but if I changed any of the parameters it would return – not even restarting UE and creating a completely new level avoided this bug. Eventually, I discovered that the issue gets fixed if I decreased the "Channel Edge Offset" to **27,000** (changed in the last step to **68,000**), but I am unsure as to what had originally caused this change or why.

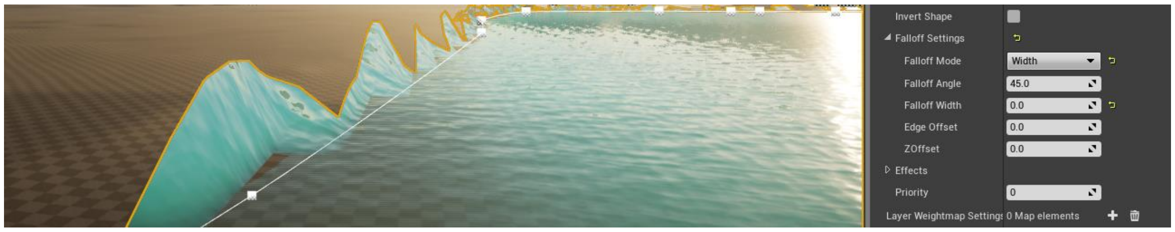




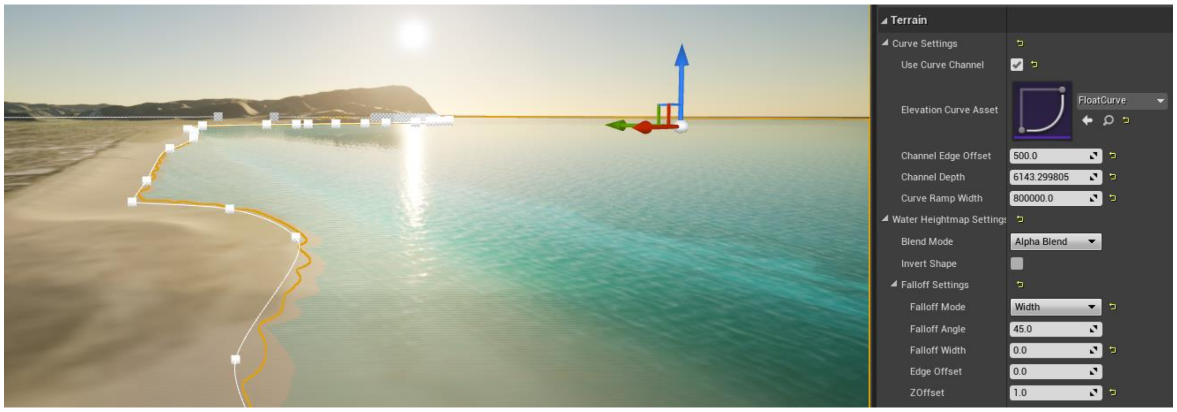




- Next, I tried adjusting the Falloff Settings. Given that I essentially wanted no falloff (as I wanted no landscape deformation in order to preserve the original landscape heights), I set all the width and offset values to zero (if "Falloff Mode" was set to "Angle", then "Falloff Angle" would need to be set to **90**), but this caused the mountainous water bug to appear again (shown in the image below) – and decreasing "Channel Edge Offset" would not solve the issue (I tested all the way to **-500,000**). However, I discovered that this was solely caused when "ZOffset" was decreased to 0, and so when I set it to **1,** I could now decrease "Channel Edge Offset" to solve the issue - which is now set to **500** (final results shown in the last image).

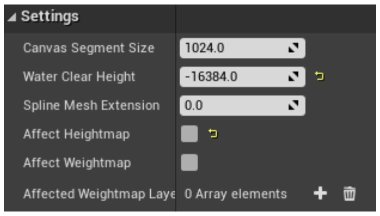


Invert Shape	☐
Falloff Settings	
Falloff Mode	Width
Falloff Angle	45.0
Falloff Width	0.0
Edge Offset	0.0
ZOffset	0.0
Effects	
Priority	0
Layer Weightmap Settings	0 Map elements



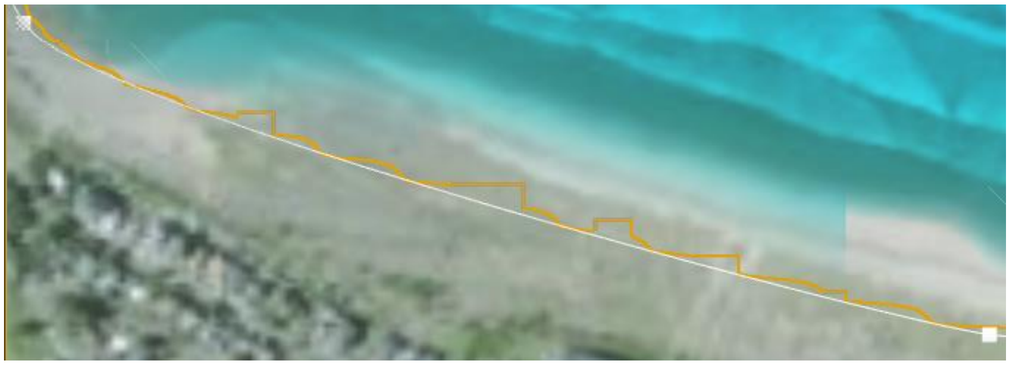
Terrain	
Curve Settings	
Use Curve Channel	✓
Elevation Curve Asset	FloatCurve
Channel Edge Offset	500.0
Channel Depth	6143.299805
Curve Ramp Width	800000.0
Water Heightmap Settings	
Blend Mode	Alpha Blend
Invert Shape	☐
Falloff Settings	
Falloff Mode	Width
Falloff Angle	45.0
Falloff Width	0.0
Edge Offset	0.0
ZOffset	1.0

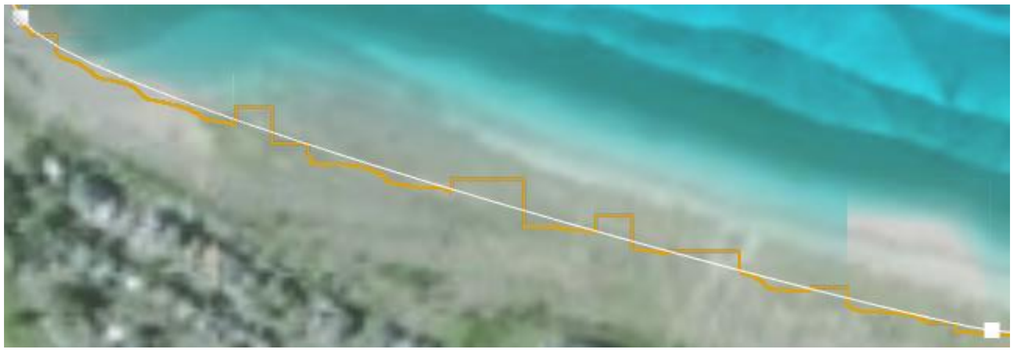
- I later discovered that, in the WaterBrushActor for the landscape, there is a "Affect Heightmap" setting that is checked. When I unchecked it (shown below), all the deformations on the landscape disappeared *without* the bug of the water mesh disappearing, which was exactly what I was after (the pixilation of the shoreline still existed, however, which can be seen in the images in the next step). After this action, changing the parameters I adjusted earlier does not visibly change anything, and performance in UE increased.



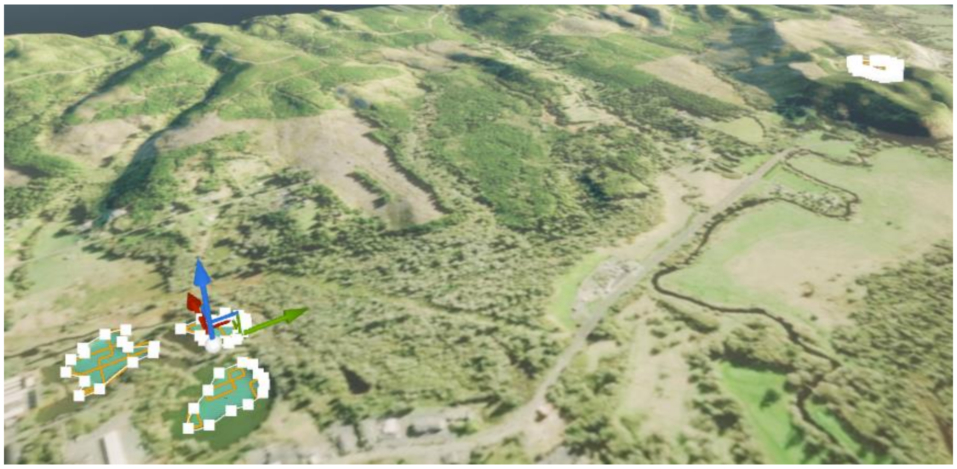
Settings	
Canvas Segment Size	1024.0
Water Clear Height	-16384.0
Spline Mesh Extension	0.0
Affect Heightmap	☐
Affect Weightmap	☐
Affected Weightmap Laye	0 Array elements

- Finally, I tried adding this WaterBodyOcean to a lower resolution landscape (for the sake of improving performance later in the project). The transition was successful with no errors (to my knowledge - all I did was copy and paste the WaterBodyOcean), except for a slight increase in pixilation at the shoreline (shown in the images below: the first shows the original terrain I was using, while the second shows the lower resolution terrain):



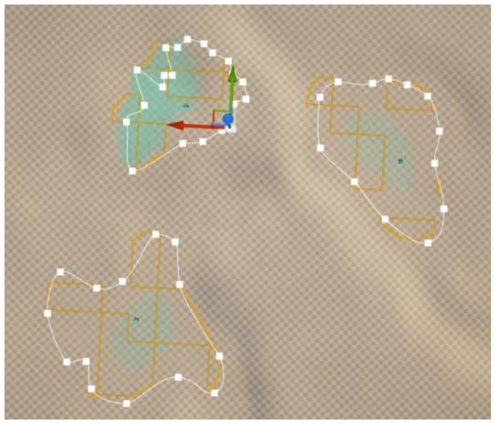


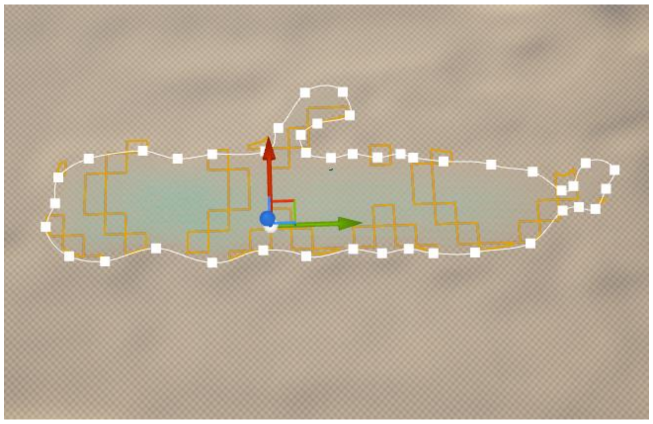
ii) After I finished with the ocean, I then added the lakes to the landscape following the same spline-drawing process as before (except I didn't have to modify their collision extents at all), shown in the images below:

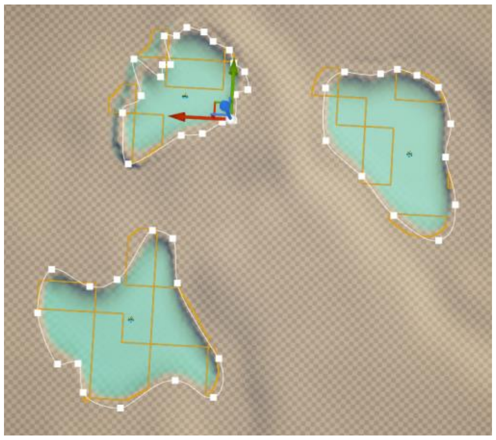


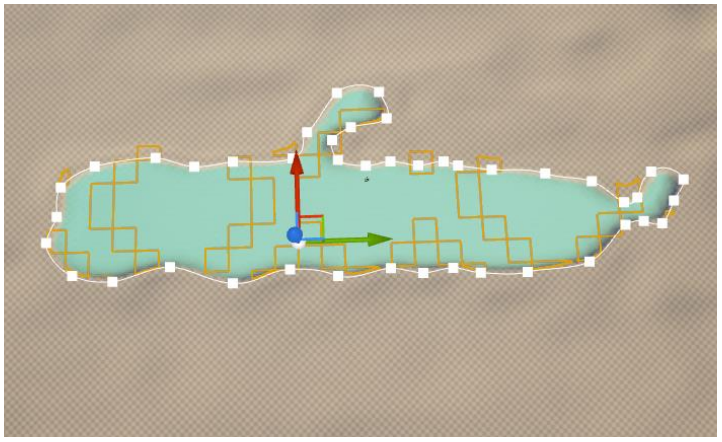
However, just like with the ocean, issues introduced themselves with these lakes as well…

- Firstly, it was clear that the water mesh did not follow the splines as closely as it did with the ocean. This could be because these lakes are significantly smaller than the ocean and so their precision is going to be less, and the checkered collision boundaries (the orange lines in the images below) seem to confirm this theory. However, when I rechecked "Affect Heightmap" in the WaterBrushActor (which I unchecked when testing the ocean – I am doing this because this actor controls the water for all water actors), this fixed the lake boundaries (as seen when comparing the top images to their corresponding bottom images) without changing the collision boundaries, and so I don't think lake size is the sole factor in this issue here (the fact that all these lakes are different sizes supports this claim). Unsurprisingly, when unchecking either "Use Curve Channel" or "Affects Landscape" for each of the WaterBodyLakes the same issue with the WaterBodyOcean appeared where the water mesh completely disappears.

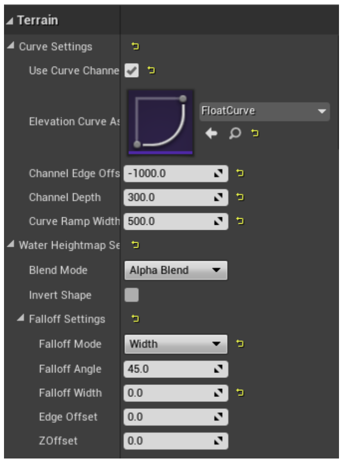




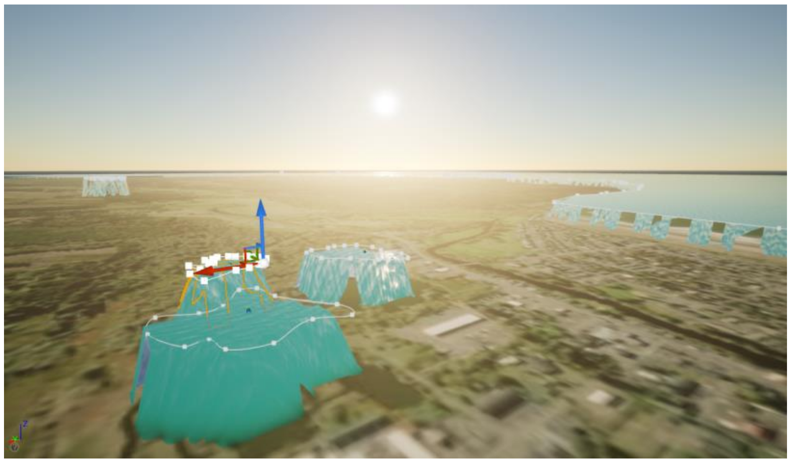


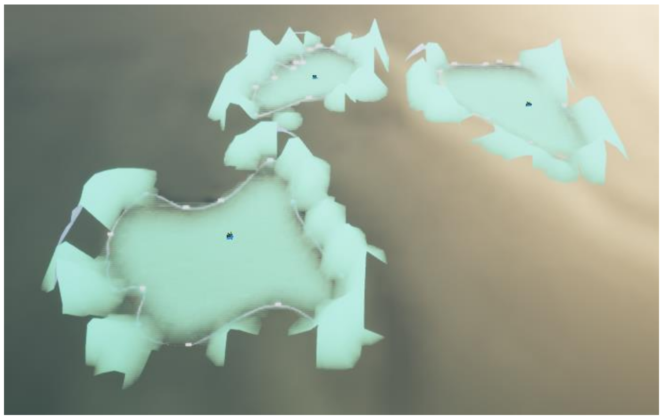


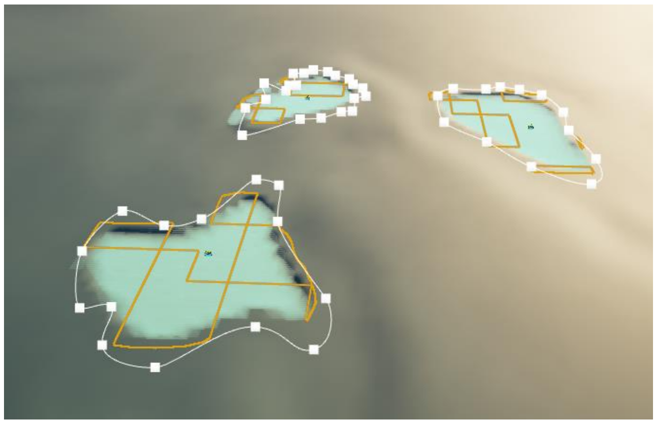
- The terrain deformation caused by the lakes wasn't an issue for me as I'm primary focused on the ocean (for the tsunami simulation) and because it fixes the lake boundary issue, and so I kept them as they are. Next, I added these lakes to the lower resolution landscape, but this instantly produced the same mountainous water issue that I got when editing the Curve Settings for the WaterBodyOcean (see the comparison between the last images below - this issue would still happen even if I unchecked "Affect Landscape" in the WateBrushActor)… luckily, also like with the ocean, changing the "Channel Edge Offset" and "Channel Depth" solved the issue (but still didn't fix the pixilation issue), using the values in the first image below – but as I was testing for the right values, a bug would come up where the entire landscape would just flatten (shown in the second image), and undoing the last edit that caused that change would not fix the issue. To get around this, I ended up reseting all of the lakes back to their original values and then only adjusting the values **once** to the values below, and didn't touch them after that. Finally, I added all these adjustments (except for the "Channel Edge Offset" change, which I kept that at **0**) to the lakes in the mid-resolution landscape for the sake of consistency.

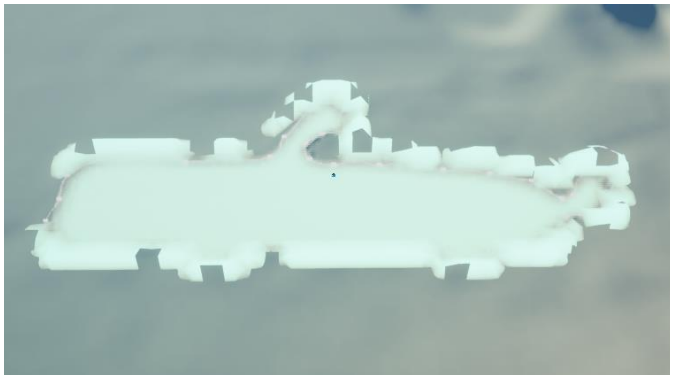


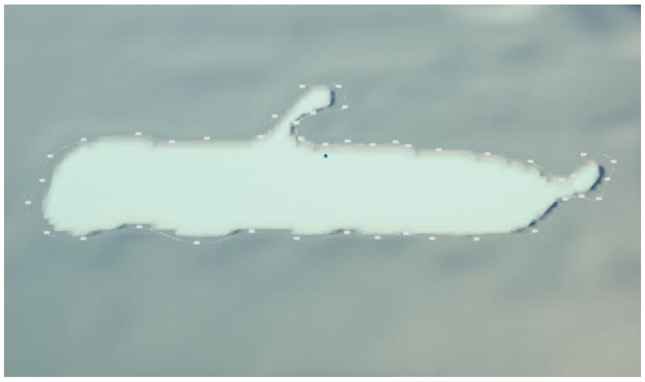
```
Terrain
  Curve Settings
    Use Curve Channe        [✓]
    Elevation Curve As      FloatCurve
    Channel Edge Offs       -1000.0
    Channel Depth           300.0
    Curve Ramp Width        500.0
  Water Heightmap Se
    Blend Mode              Alpha Blend
    Invert Shape            [ ]
    Falloff Settings
      Falloff Mode          Width
      Falloff Angle         45.0
      Falloff Width         0.0
      Edge Offset           0.0
      ZOffset               0.0
```





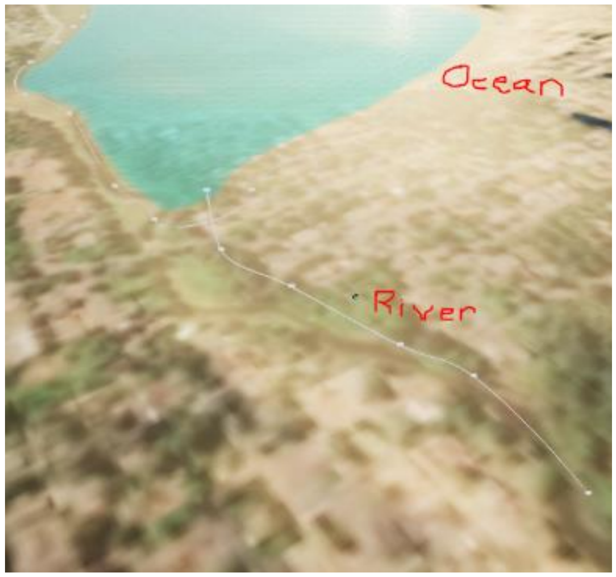


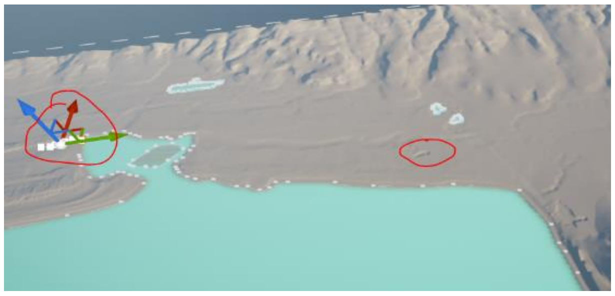


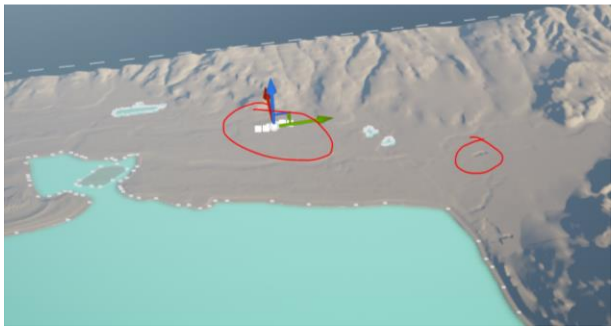


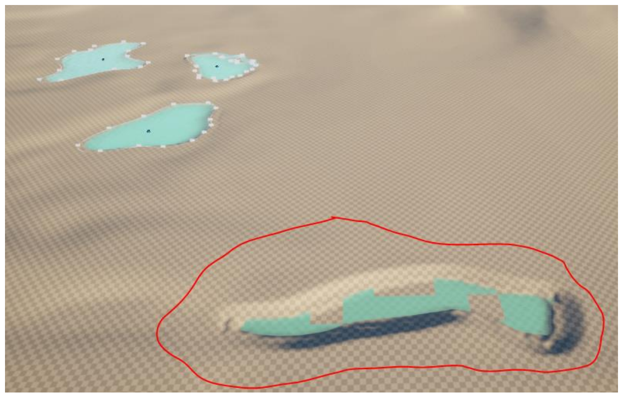
iii) Lastly, I am currently working on adding rivers to the landscape. With this water body, however, issues immediately became apparent, mainly with its connection with the water mesh…

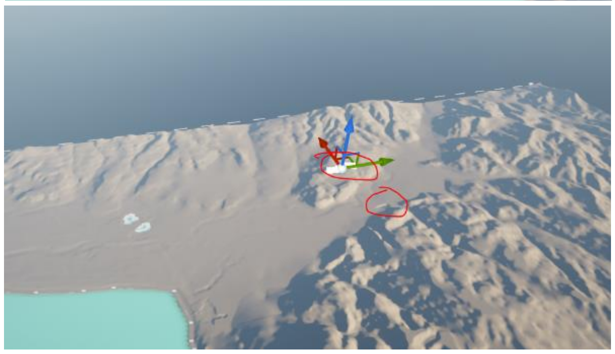
- The first thing I did was attempt to insert a WaterBodyRiver and trace out a small river on the landscape. However, it failed to perform any noticeable deformation to the landscape (first left image below). In fact, the main difference I noticed was that UE began heavily lagging, to the point hat I had to restart the software and reload the level. It didn't unusually lag after that, but anytime I modified the WaterBodyRiver at all it returned. A couple days later after I discovered a way to avoid this issue, I realized that the river *was* deforming the landscape, but not in the area that it was supposed to (second left image below)... and interestingly, the relationship between the two positions was not linear! If I moved the WaterBodyRiver to the right, so too would the landscape deformation but at a slower speed (right images below), and there seems to no significance at the position at which they meet. I tried this on the low-resolution landscape as well, which produced the same issue. I am unsure of what's causing this problem or how to fix it.



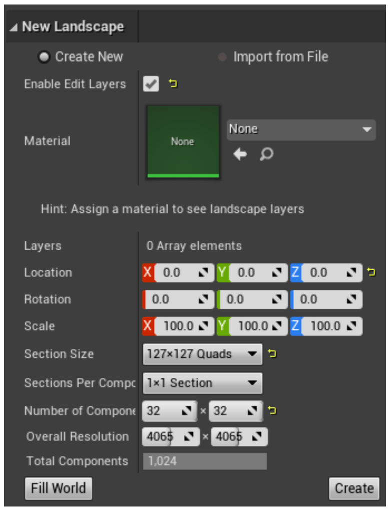








- I first suspected that this bug has to do something with the landscape resolution, but given that it made the same issue on the low-resolution landscape. I also tried this with the full resolution landscape (which automatically cropped it to **8161**x**8161**), but then UE crashed with a "Out of video memory trying to allocate a texture" error (it recommends that the resolution be lowered). Next, I suspected that this has to do with the number of components a landscape is composed of, but that is clearly not the case as I was successfully able to use rivers (as well as oceans, lakes, and islands) on the default UE landscape with **1,024** components (shown in the images below – it's also important to point out that the collision boundaries of these actors look normal, unlike the collision boundary for the lakes earlier).

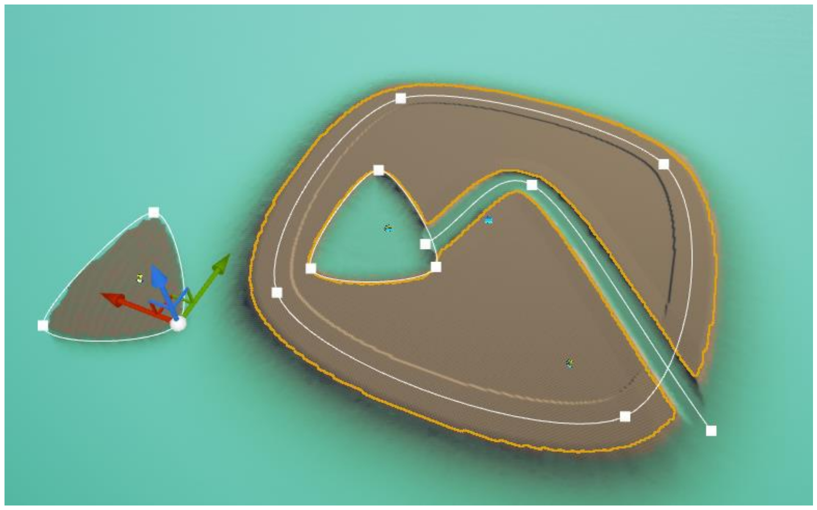


New Landscape

● Create New ○ Import from File

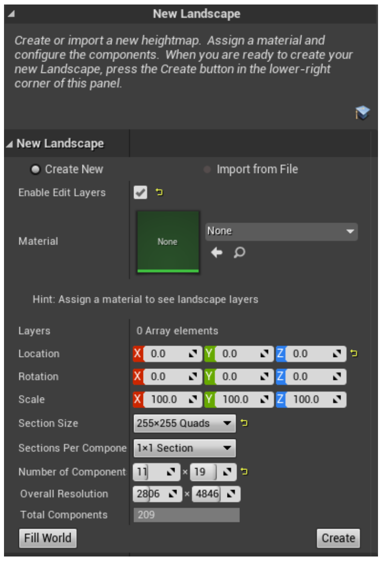
Enable Edit Layers	✓
Material	None
	Hint: Assign a material to see landscape layers
Layers	0 Array elements
Location	X 0.0 Y 0.0 Z 0.0
Rotation	0.0 0.0 0.0
Scale	X 100.0 Y 100.0 Z 100.0
Section Size	127×127 Quads
Sections Per Compo	1×1 Section
Number of Compone	32 × 32
Overall Resolution	4065 × 4065
Total Components	1,024

Fill World Create



- I was attempting to solve an entirely different issue when I accidentally discovered a case in which rivers do work: for the mid-resolution landscape we're using, UE defaults to **12**x**20** components. When I gradually decreased this to **11**x**16** components (which crops the landscape according to the new dimensions), WaterBodyRivers immediately began working as they should without the intense lag from before - any number higher than 16 will cause the bug. Interestingly, at **11**x**8** components, a different kind of bug appears where these small hills extend from the river's ending position, which gradually become farther apart until **11**x**1** components, where the first hill is almost unnoticeable from how far away it is (shown in the images below). I also tried starting from **10**x**17** and gradually decreased to **1**x**17**, but for each case the original bug kept appearing (although, the intense lag suddenly disappeared after **8**x**17** components) …. This made me theorize that the relationship between the X and Y components isn't the problem, but rather the individual values themselves. I tested this out by creating a **16**x**16** landscape (as problem seemed to happen when one of the component axes was greater or equal to **17**), and this allowed for the successful creation of a river! I also tried with other combinations where one of the values was **16** or lower while the other was **8** or lower, and the second (below) bug appeared each time. I also tested with component sizes where both values were between **9** and **16**, and rivers were successfully created each time. As such, I am confident that these bugs are caused by the number of components along a single axis rather than the total number of components. However, there seems to be another factor at play here, as I tried this with a **17**x**1** landscape, which created a successful river… Furthermore, I tried with a **17**x**16** landscape, which oddly produced the second bug (as well as some lag, but the **16**x**16** one lagged as well, so I'm assuming that this is due to the limitations of my machine rather than an actual bug) … As such, it appears to me that these bugs are mainly caused by the number of components along one axis, as well as some other factor that escapes my understanding.

- I tried a portion of the above tests with the low-res landscape, and the same results occurred (without the intense lag with failed rivers). I also tried adding a river to a random image I found, which defaulted to both component values being above **20**. Unlike the previous tests, however, a successful river was added. This caused me to assume that the heightmap image itself plays a role in this bug… however, when I tried on a blank landscape with the same dimensions as the mid-resolution heightmap (settings shown below), the same bugs appeared again! At this point, this bug has become so complex that it has taken too much of my time, so I chose to use the cropped low-resolution landscape with component dimensions of **13**x**16** (closest to original values [**14**x**23**] without losing too much of the landscape).



New Landscape

Create or import a new heightmap. Assign a material and configure the components. When you are ready to create your new Landscape, press the Create button in the lower-right corner of this panel.

New Landscape

[✓ Create New] Import from File

Enable Edit Layers	✓
Material	None
Hint: Assign a material to see landscape layers	
Layers	0 Array elements
Location	X 0.0 Y 0.0 Z 0.0
Rotation	X 0.0 Y 0.0 Z 0.0
Scale	X 100.0 Y 100.0 Z 100.0
Section Size	255×255 Quads
Sections Per Compone	1×1 Section
Number of Component	11 × 19
Overall Resolution	2806 × 4846
Total Components	209

Fill World   Create

- I am currently at the step of drawing out the rivers so that they accurately match the satellite image and natural indentations in the heightmap. The main issue at the moment is getting the right height added to each spline point, as the plugin defaults to linear interpolation between the first and last points (so if the first point has a height of 100 with the last point at a height of 0 [both UE automatically finds using the landscape height at those points], and I create a point in between them, UE will default to making its height at 50 – even if the landscape's height at that point is something else).

iv) Using WaterBodyCustom (I'm hopeful I can simulate a tsunami with this)