Frogger Styled Game Instructional Guide

This video is a quick reel of some of my work from previous projects. If you want to know more, feel free to message me and ask questions.

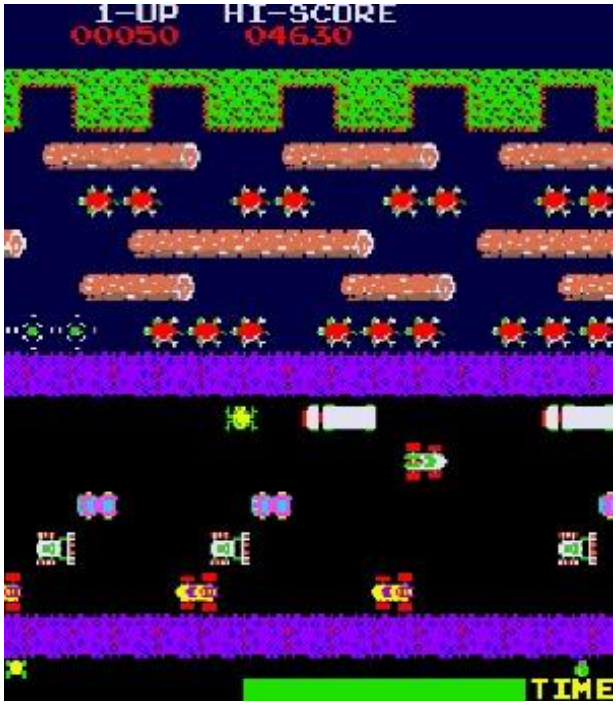
Overview

I'm sure those of you who are reading this instructional guide have dreams of entering the gaming industry whether as a game programmer, designer, artist or maybe as an indie developer. As you move along the path of life chasing your dream you will see that the process of making games can be long and difficult, therefore, Full Sail and I will help prepare you for your future in game design. While you may dream of making the next Call of Duty or Fortnite it will be much more beneficial to start small with a simple Frogger inspired game using Unreal Engine 4.26.2. In this document I will provide step-by-step instructions on how to make a game. Once you are done, you should have a Frogger inspired game that you can add your own spin to, and hopefully you'll have the confidence and knowledge to experiment and make your own games in the future.

Instructions

Before we begin, let's go over the goal of Frogger. When you start the game, you are given 3 to 7 frogs that act as lives. Your goal is to guide a frog to an empty spot at the top of the screen called a "frog home" while avoiding obstacles like cars and snakes. Losing all the frogs will end the game. You can hop one spot at a time in four directions. There are also opportunities, like with many arcade games, to earn points and get a high score. With this information in mind, we can now make our own game with similar mechanics but with a different theme and style. For this project I will utilize a more robot/futuristic style but once we're done you will have the skill to choose and implement a different style.

Frogger Styled Game Instructional Guide

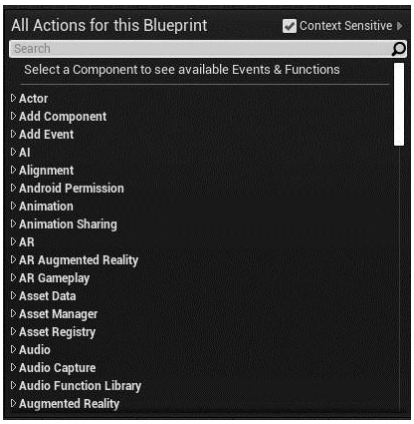


General Information

Throughout the instructions I will ask you to add in multiple types of nodes. I will go over how to do that in this section as a refresher to avoid constantly explaining the process.

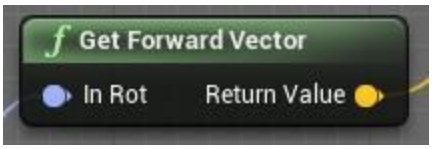
Adding in Nodes

- One way to add in nodes and events is to right click in the Event Graph which will bring up the context menu. From there you can either use the search bar or look at the sections to find the desired node, event, or function you want to add.



- You can also get the context menu by dragging off the output pin or by returning value a node and then releasing.

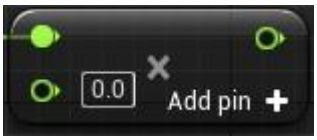
Frogger Styled Game Instructional Guide



- You can also increment a float or integer value type ++ in the context menu to get an increment node.

- To do addition, then type + and to do subtraction, then type in a -

- To do multiplication, then type * and if you want to do division type /.

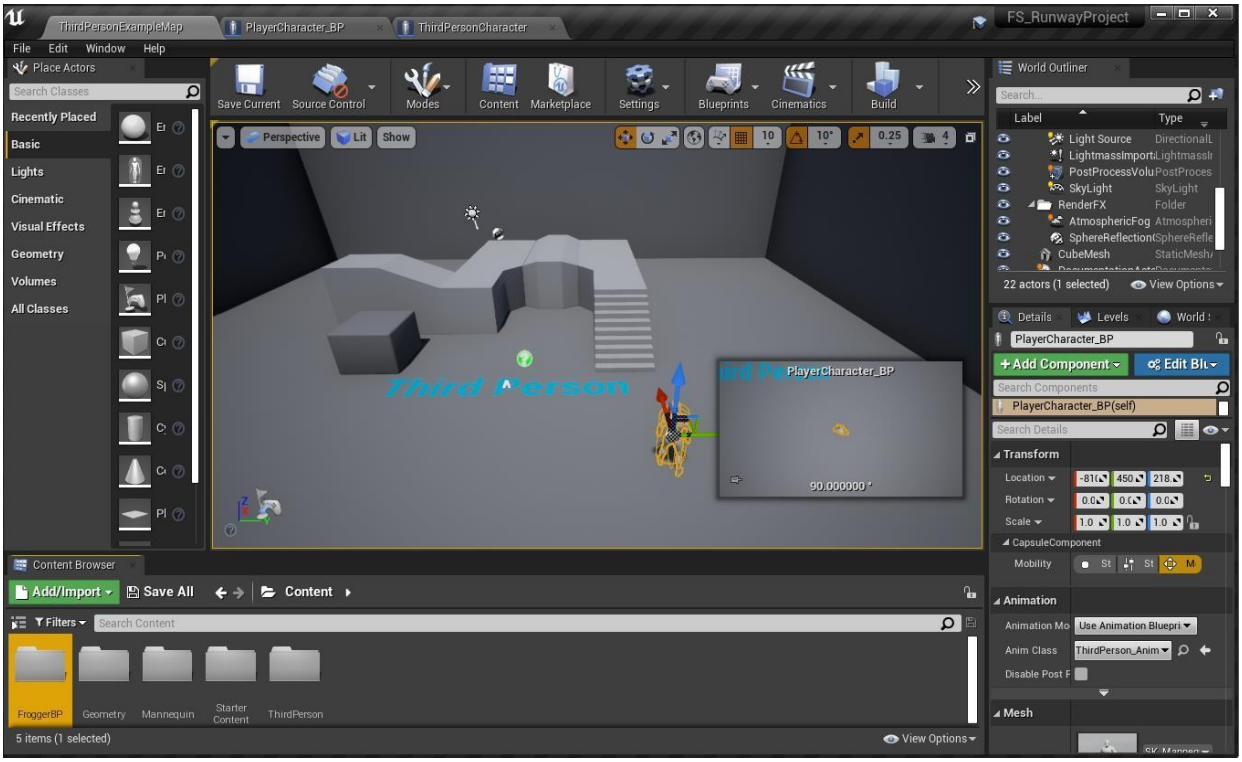


- To check if a value is equal to something, then type =.

Part 1: Setting up the Player Character

For this project I will use Unreal Engine's Third Person template with starter content. Once you open the Unreal Editor your screen will look like this.

Frogger Styled Game Instructional Guide



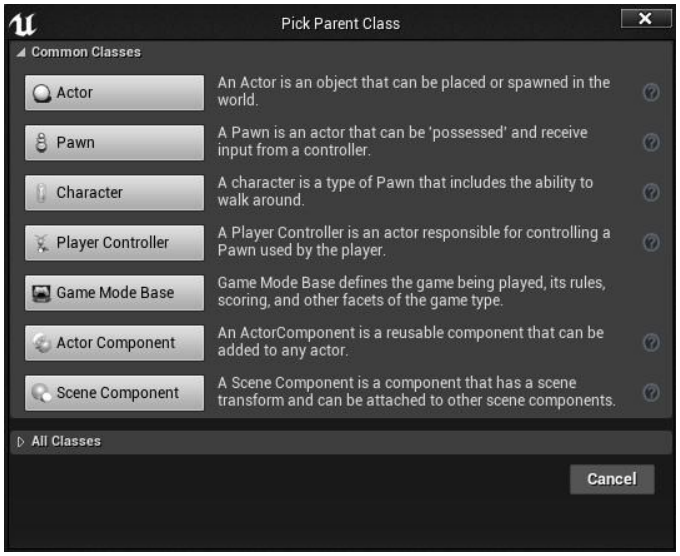
Making the player Character

Let's start by making the player character.

- In the Content Browser click on the folder called ThirdPersonBP which I renamed FroggerBP to open it.
- Then click on the Folder called Blueprints. This will show the Third person character blueprint class and the game mode blueprint class.
- Because Frogger moves differently compared to modern game characters we are going to make our own playable character with its own unique movement. To do this either right click in the empty space in the Content Browser or click the green Add/Import button. Once you do, you should see a menu like this.

Frogger Styled Game Instructional Guide

- Then you are going to click the Blueprint Asset button which will bring up another menu that looks like this.



- Since we are making a player character, we are going to select the character parent class, and I'm going to name our character PlayerCharacter_BP, but you can select to name it something else.
- For this next step you can choose to either use the third person character Mannequin or make your own model for the player character. I'll detail both ways so feel free to choose which steps you want to follow to make the player character. I highly recommend making and using your own model.

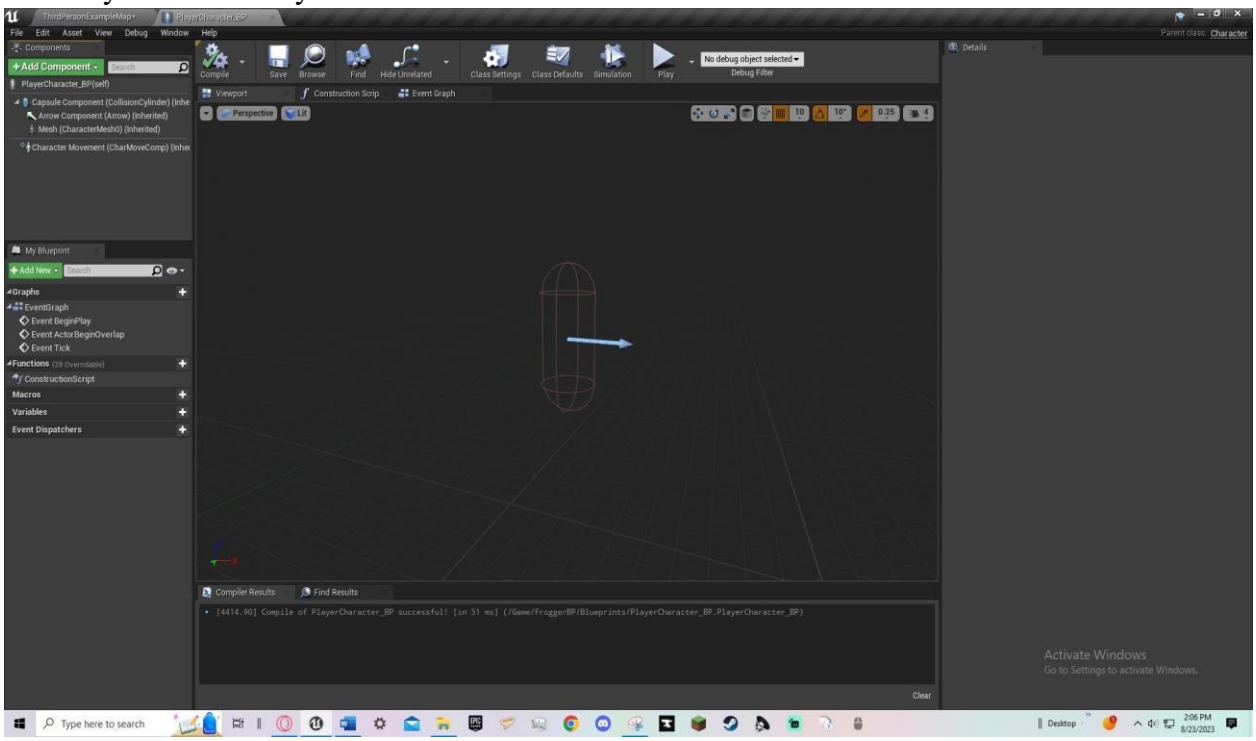
Frogger Styled Game Instructional Guide

Provided Mannequin

Adding and Setting Up the Mannequin

Now we are going to add in the player mannequin and set it up for the future implementation of the camera and movement.

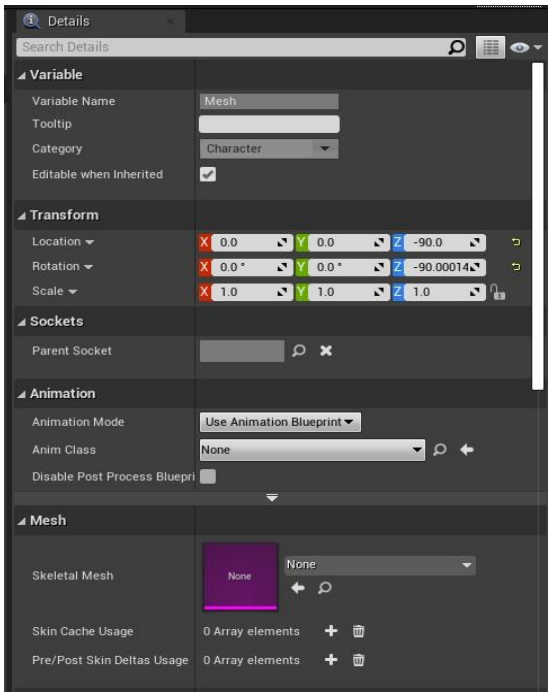
- In the content browser click on our newly created character to edit and make changes to it. Once you click on it your screen should look like this.



- To add the character mesh, you must click on the mesh component found at the top left of the screen which I circled in the picture below.

Frogger Styled Game Instructional Guide

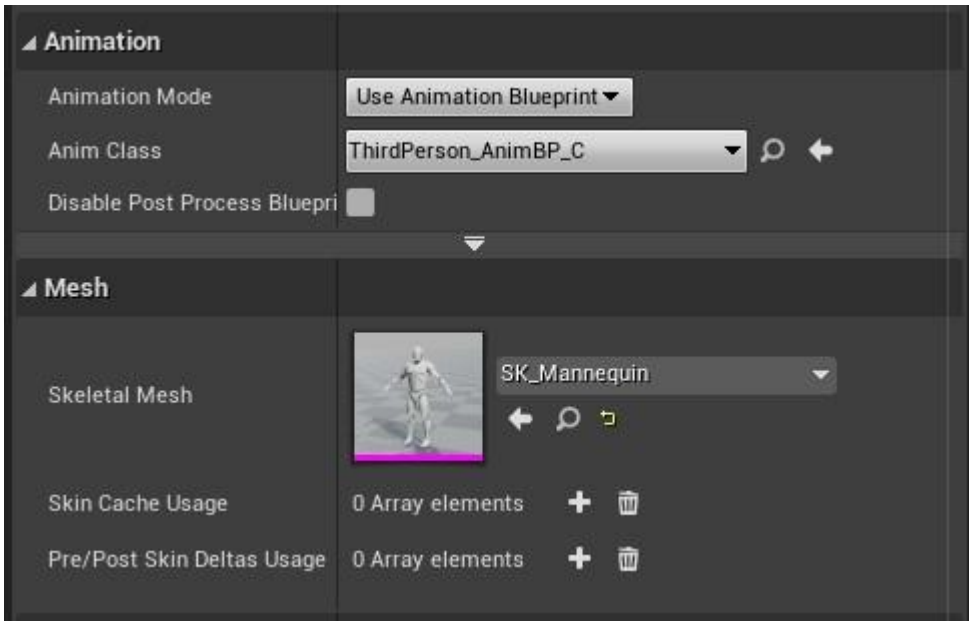
Once you click on the mesh component you will see information appear on the right side of the screen in the details tab which will look like this.



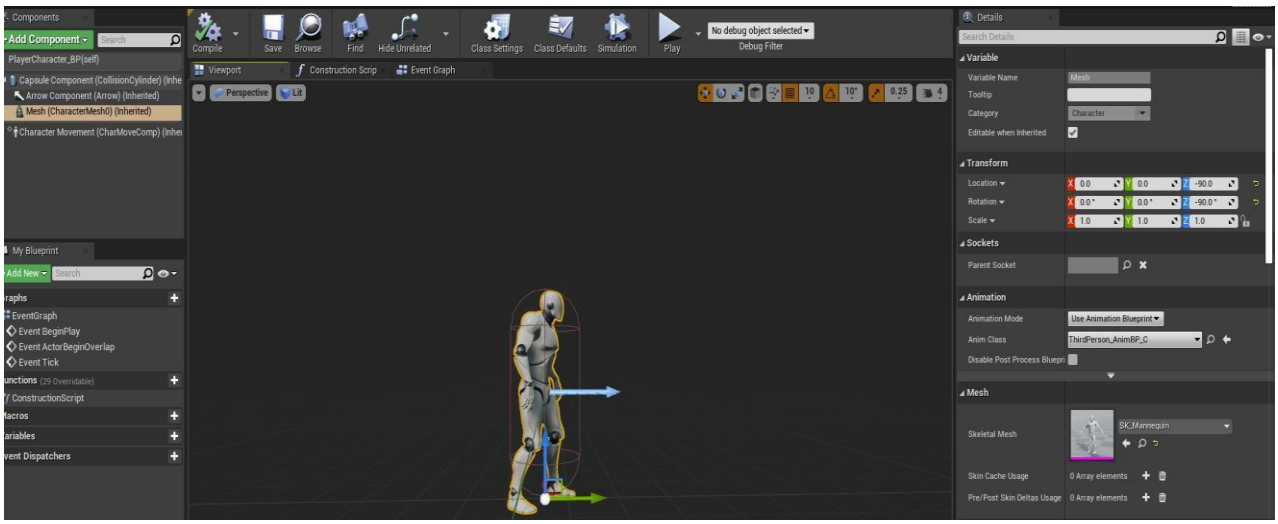
- Under the mesh section there will be A skeletal Mesh with a selector. Choose either the SK_Mannequin or SK_Mannequin_Female.
- Once you have a mannequin you will now have to move and rotate the mannequin to make sure it's facing the right direction.
- To move the mannequin, you can either you the multicolor arrows or type in numbers in the Location and Rotations parts under the Transform section.
- Regardless of the method you choose make sure that the mannequin is within the capsule and facing the same way as the blue arrow.
- Do not type anything into the scale as that will change the size of the mannequin, which we are not doing now.
- If you want to type in the values instead of using the arrows, then the Z location should be at -90 and the Z rotation should also be at -90.

- After you have the mannequin in place you now have to give it an animation class which can simply be done by going to the animation section on the details tab, clicking the drop-down menu on the Anim Class, and then choose the ThirdPerson_AnimBP_C.

Frogger Styled Game Instructional Guide



- Once done the mannequin should look like this and now, we are ready to add the camera.

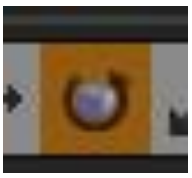
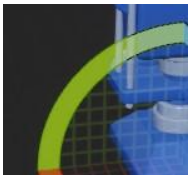


Adding the Camera (Model & Mannequin)

- To add the camera, click on the green Add Component button and either type camera in the search bar or scroll down until you see it. Click on camera and it will be added to the player character.
- Now that it has been added, we must move it in place to create a top-down view. To do this set the camera's X and Y position to 0 to center the camera on the player and then set

the Z value to 900. Feel Free to change the camera locations to whatever looks best as long as the camera is top-down.

Now we are going to set the camera's Y Rotation to -90 degrees by either typing the value in or by selecting the rotation option, clicking on the green segment, and dragging it.

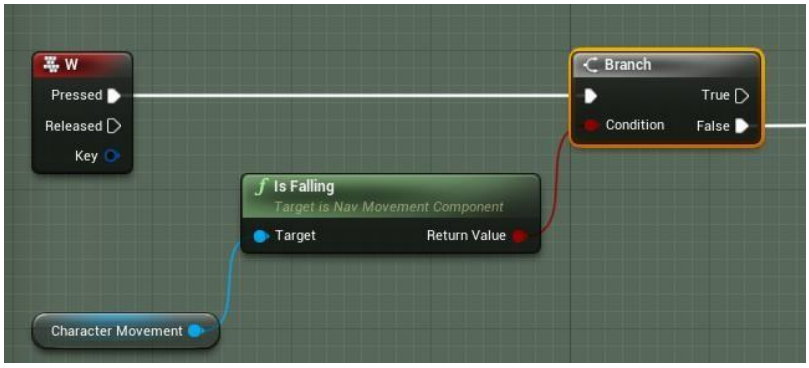



- If you press play you will see that there is now a top-down view, and we proceed to making the character move.

Making the Character Move (Model & Mannequin)

- Go into the Event Graph and add in an Input W event node, and you'll get an event node that looks like this. This event will handle forward movement for our character.

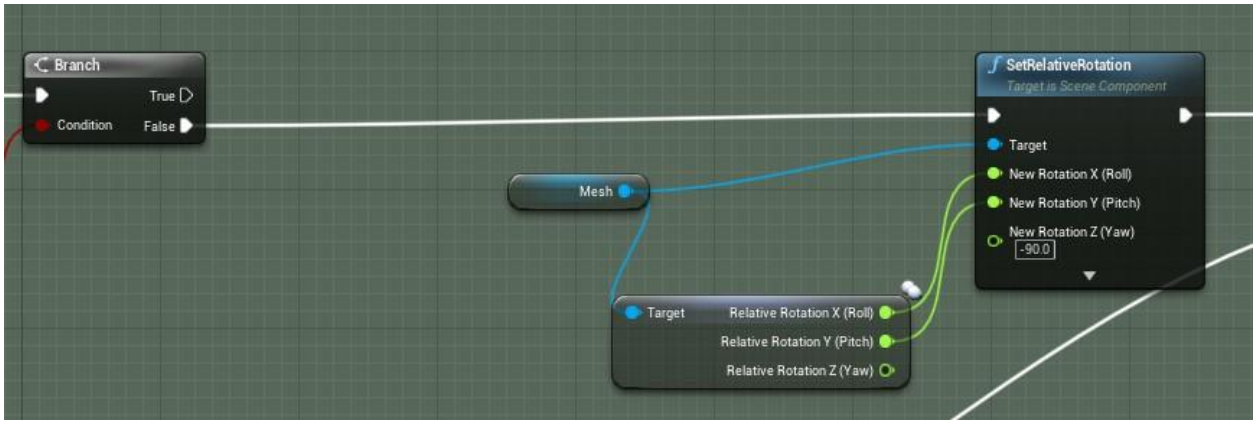
- Then in the component section click and drag the Character Movement component into the event graph. Connect the Character Movement component to an "Is Falling node" and then add in a Branch node and connect it to the return value of the" Is Falling node". We're using a Branch node so that the player character can't do multiple jumps in the air. Connect the W event and the Branch node and once done it the code should look like this.



- In the component drag in the Mesh component and connect it to the target pin of a Set Relative Rotation nide. Connect the Set Relative Rotation node to the False Exec pin on the Branch node.
- Drag off the Mesh component and get its relative rotation. Right-click the relative rotation pin and split the struct pin. Connect the X and Y rotations to the X and Y pins of the Set

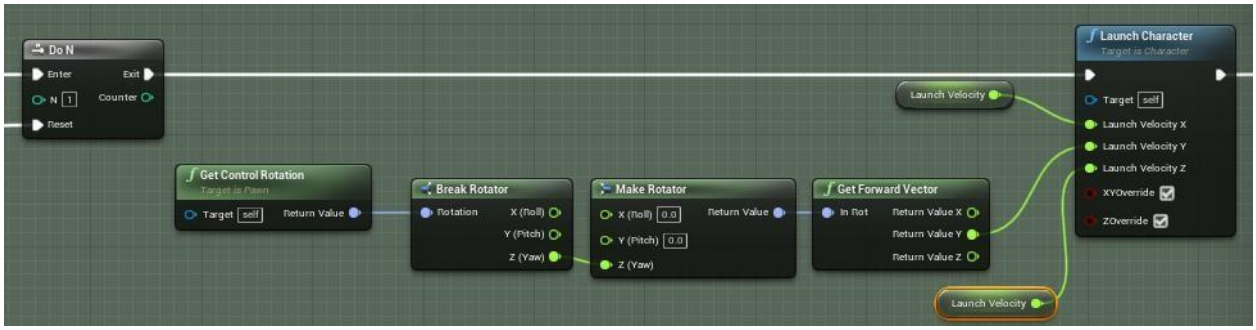
Frogger Styled Game Instructional Guide

- Relative Rotation node. For the Z rotation pin of the Set Relative Rotation node, we are going to type in -90. This will ensure that our character is facing forward when we go forward. Once done the code should look like this.



- Now we are going to implement the movement which works the same way if use a mannequin or make your own model. First add in a Do N node.
- For the N value make it 1 which determines the number of times our character will jump.
- Connect the Do N node to the Set Relative Rotation node.
- Connect a Launch Character node to the Do N node's Exit pin.
- Next, we are going to add in a Get Controller Rotation node.
- Drag off the return value of the Get Controller Rotation node and connect it to a Break Rotator.
- Drag off the Z (Yaw) value and connect it to a Make Rotator.
- Drag off the Return Value pin from the Make Rotator node and connect it to a Get Forward Vector node.
- Next, we are going to make two float variables called Launch Velocity and Negative Launch Velocity which we will use for our launch character node. For the Launch Velocity I changed the default value to 300 and -300 for the Negative Launch Velocity. I found that these values worked best for me, but you can change it later if you want your character to go further.
- Drag in the Launch Velocity variable and connect it to the launch velocity X and Y pins.
- Then click on the X and Y override boxes on the launch character node.
- One done the code should look like this.

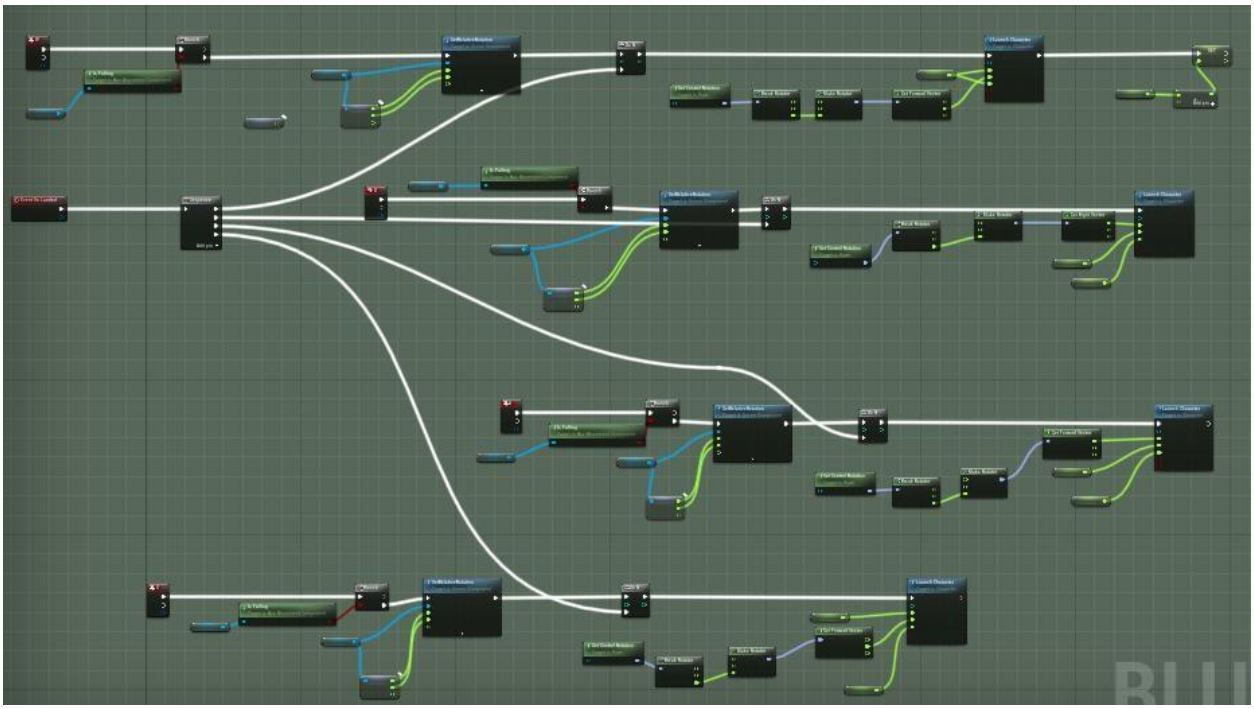
Frogger Styled Game Instructional Guide



- Now that we have the forward movement set up, we are going to set up the movement for the A, S, D.
- In the event graph right click and type in Event on Landed to get the node.
- Connect a Sequence Node to the exec pin on the Event On Landed node.

  This sequence node will allow us to move in multiple directions. Add three more pins to this sequence node.
- Add the input events for the A, S, D keys the same way we added the W input key.
- The way to make the characters move is the same way as we did before but with slight differences.
- Copy and paste the code we used for W key input three times and connect it to each new key input event.
- For the D key input event set the New Z rotation to 0 and on the Set Relative Rotation node keep the Launch Character node the same as for the W key.
- For the A key input event set the New Z rotation to 180 and on the Set Relative Rotation node plug in the Negative launch Velocity into the launch velocity Y on the Launch Character node and the Launch Velocity variable into the Z.
- For the S key input event set the New Z rotation to 90 and on the Set Relative Rotation node plug in the Negative launch Velocity into the launch velocity X on the Launch Character node and the Launch Velocity variable into the Z.
- Next, we are going to take the Then Exec and plug it into the Reset Exec on the Do N node. Each Do N node should have only one Then Exec plugged into it. Once done the code should look like this.

Frogger Styled Game Instructional Guide

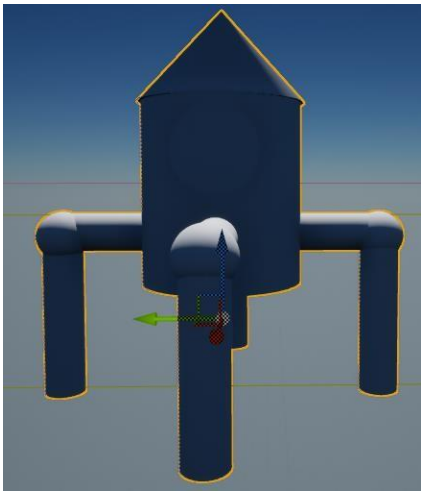


Custom Model

Making and Setting Up the Model

- If you don’t want to use the mannequin that Unreal Engine has then we can use a static mesh instead. If you do want to use a static mesh, then make sure it’s around the size of the third person mannequin.
- I recommend using static mesh shapes found in the basic section to make a character that I made.
- I suggest making a folder for our static mesh and for any future meshes we make.

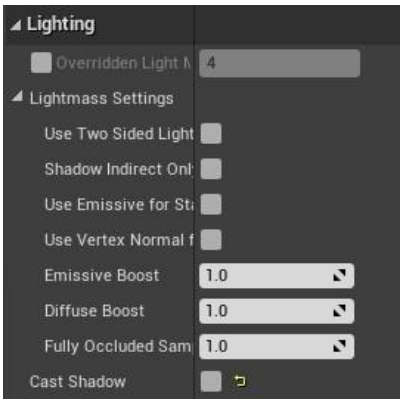
Frogger Styled Game Instructional Guide



- Once you have a character made select all the pieces you used to make it, right click, and then click group to move the whole character around in case you want to add more or change it, and then right click and then click on covert actors to static mesh.

- If you get a red error mesh that looks like this.

- You can either rebuild lighting or go to whatever static mesh you placed in and in the details tab under the lighting section uncheck cast shadow. This will make the message go away.



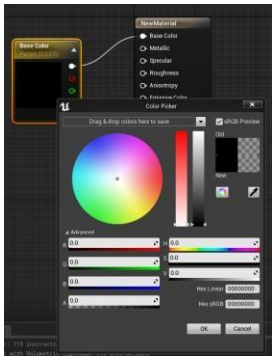
Frogger Styled Game Instructional Guide

- Choose a folder to send it to, name the mesh and then click OK to create it.
- I have instructions further down if you want to know how to make different colors to use for our model.
- Use the same steps to make the character that we went through previously but for the character mesh leave it blank.
- To add our model, click on add component, choose static mesh, and then choose our character model mesh.
- Attach the static mesh to the character mesh which will allow us to make our static mesh model jump and move with the launch character node.
- To set up character movement and the top-down camera follow the same steps I listed previously.

Making and Adding Materials/Colors

In this section I'm going to go over how to make materials to use as colors.

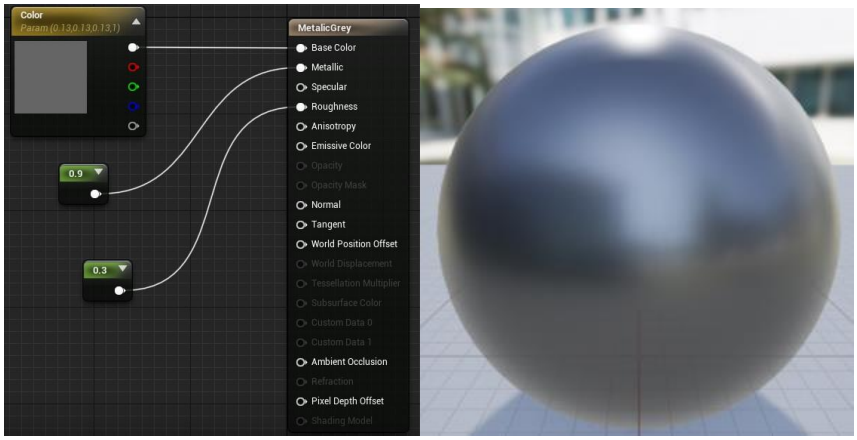
- I recommend making a folder for materials. Either right click or click the Add/Important and then click on Material.
- Open the new material folder we made, and you will see a NewMaterial return node with multiple pins.
- Right click the base color pin and promote it to a parameter to change the materials color.
- Double click the Base Color parameter to go into the color picker and pick color.



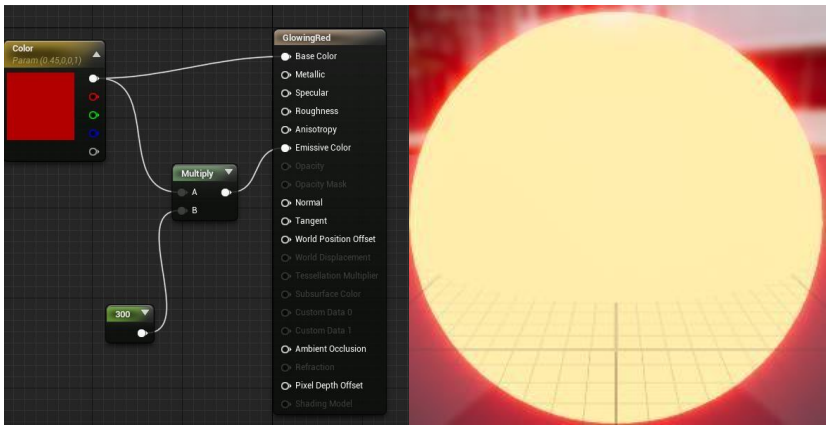
- I you want the material to look metallic or rougher, then right click and type constant to get a node like this. Click on the node and choose a value for it.

Frogger Styled Game Instructional Guide

- Make multiple of these if you want to make multiple parameters.
- Here's an example of a metallic grey material.

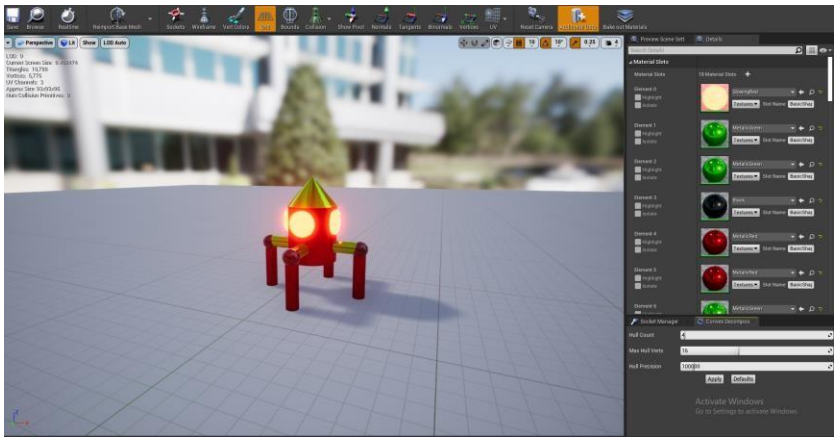


- If you want to make a glowing material, add in a Multiply expression node. Plug the white pin on the base color into A, plug a constant into B and then drag the pin on the right of the Multiply expression into emissive color and that will make the material glow.



- If you want to add a color to a static mesh open the static mesh by directly clicking on it in the content browser to edit it. You should see a screen like this where you can select a material for each part/element of your static mesh in the material slots section of the details tab.

Frogger Styled Game Instructional Guide



- If you're using the static mesh for the player or an actor, then you can open the blueprint, click on the static mesh you're using in the components tab and there will be a materials section where you can pick a material for the different elements/parts of your static mesh.

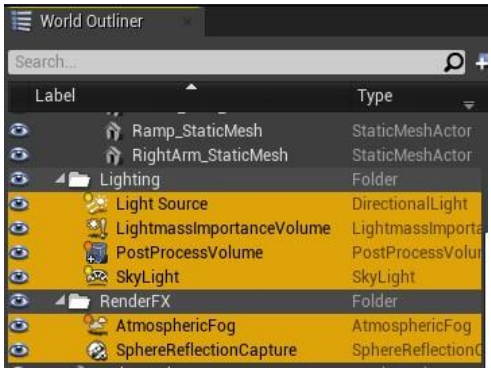
Part 2: Making Levels

Now that we have finished setting the player camera and movement, we can move on to making levels for our game. Once we're done, we will have a level template that we can use for future levels in our game.

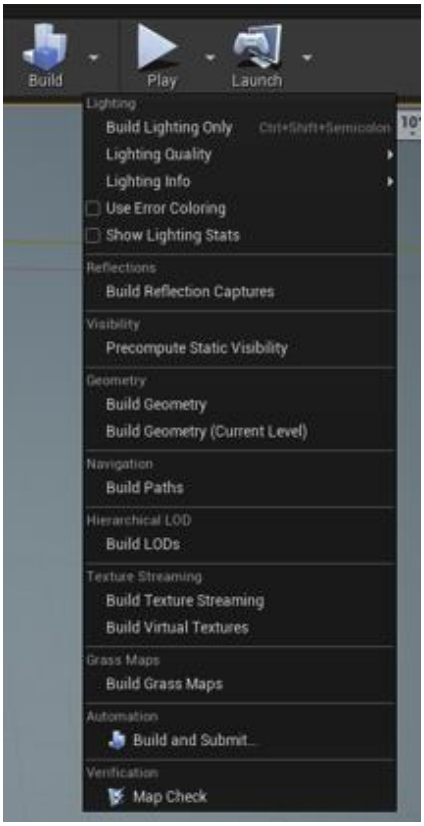
Setting Up Basic Level Lighting

- If you want to, make a folder called maps in the content browser to hold your levels.
- Right click in the content browser and click level to make a map file. I'm going to name this map Level1.
- When you open your level, the screen will be black because the level doesn't have any lighting.
- Since we are using the third person template, we can open the ThirdPersonExampleMap and copy the lighting there to paste it into our level. In the world outlier on the right of the screen there are 2 folders called Lighting and Render FX. Hold the CTRL key and click the actors under each folder. Once they are all highlighted press CTRL C to copy them.

Frogger Styled Game Instructional Guide



- Open your new level and in the level outlier and there will be something that will be whatever your level name is with (Editor) next to it.
- Click on it then hit CTRL V to paste the copied actors. A white message about rebuilding reflection captures will appear on the left of the screen.
- To make it go away click the arrow next to the build button and click build lighting only.



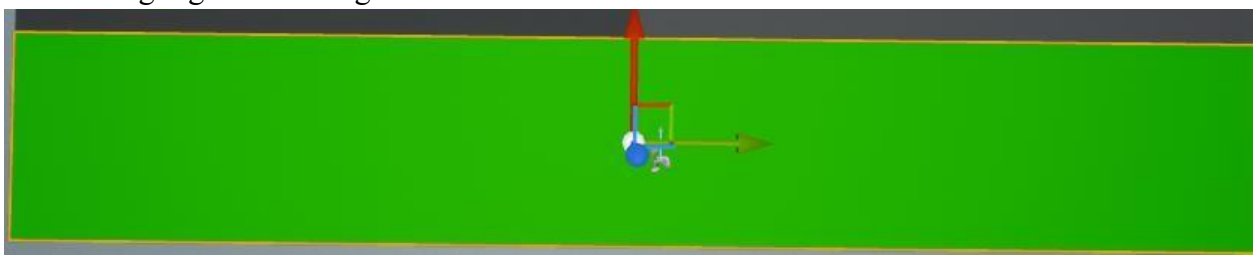
- Once the lighting is built click save current. and now our level will have lighting.
- If you want to change the color or brightness of the lighting, then click on the light source actor. The intensity option controls the brightness while the light color controls the color of the light source.

Frogger Styled Game Instructional Guide

Building the Level

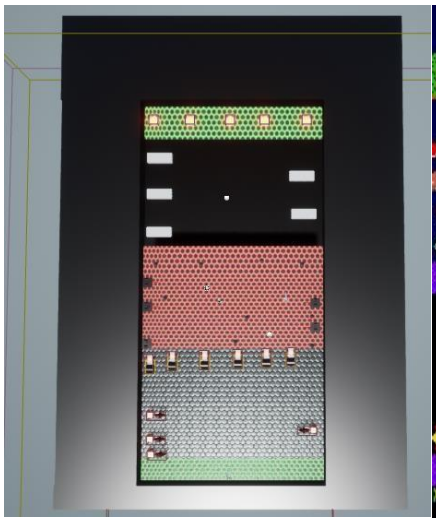
For this section you don't need to make your level look exactly like mine if you desire a different design or theme, but these steps will help you make a basic level that can fit the Frogger Gameplay.

- The first thing we are going to do is make a platform for our player character to stand on. We can do this by either dragging in a cube static mesh found under the basic section in the place actors tab or by dragging in a box brush found under the geometry section. I used a box brush because I found it easier to measure the player's jump distance to adjust it as I saw fit.
- Adjust the size of the cube or box as you see fit to make the platform. Once the platform is made drag in a Player Start actor into the level.
- Click on the Player Start actor and then click on world settings. For the Game Mode Override option choose the game mode we made, for the Default Pawn Class option choose the player character we made, and for the Player Controller Class choose the controller we made. Once we set up our player will be able to move around the level.
- If you want to make colored sections for your level, I recommend using planes which can be found in the basic section of the place actor tabs. You can place planes on top of your level platform and change the color of the plane. Here is an example of me using a plane and coloring it green with a green material I made.



- Since we added starter content to our game, we can go to the Starter Content folder in the content browser and in the folder is another folder for materials that you can use.
- I recommend adding walls to your level to keep the player from falling off.
- As I said before you can choose how you want to design your level, just make sure that it is long enough to have plenty of gameplay.
- I also suggest that you disable the cast shadow options of any static meshes you put into your level so that you don't have to constantly rebuild the lighting.
- Here is a zoomed out top-down view of my level and a Frogger level that you can use as a reference.

Frogger Styled Game Instructional Guide



- As you can see, I used multiple planes with different materials to make sections of map that also represent different gameplay beats with different obstacles and mechanics like in Frogger.
- Once you have your first level done you can make more by duplicating the first level and then make changes to the duplicate level.

Part 3: Player Systems

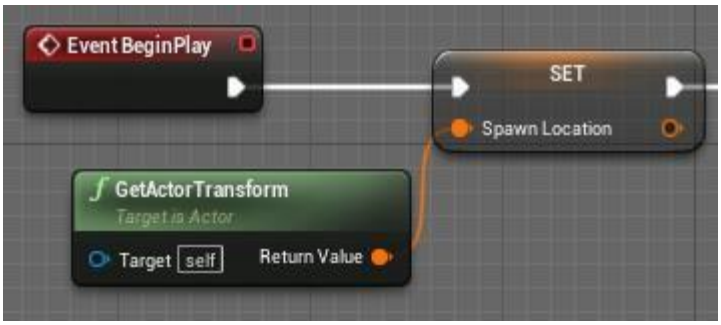
Now that we have our level made, we can implement key player systems.

Player Variables

Before we start making obstacles let's make a health and lives system for our player character and a way for our player to respawn.

- First, we are going to open our player character and make a float variable called Health, a float variable called Time Remaining, a float variable called Score, an integer variable called Lives, and a transform variable called Spawn Location.
- For the Lives variable I'm going to set the value to 3, for the Score variable set it to 0, and for health you can set the value to whatever seeing how the player is going to die in one hit so I'm just going to set it at 100.
- The Time variable represents the amount of time the player has to get to an objective. If the player doesn't reach the objective in time, then they will automatically lose a life. I'm going to set the Time variable to 60 which represents 60 seconds.
- To set the value of the Spawn Location variable drag out the variable and click set Spawn Location. Right click and type get actor transform to get the node. Connect the get actor transform return value pin to our Set Spawn Location node and then Connect this node to Event Begin Play.

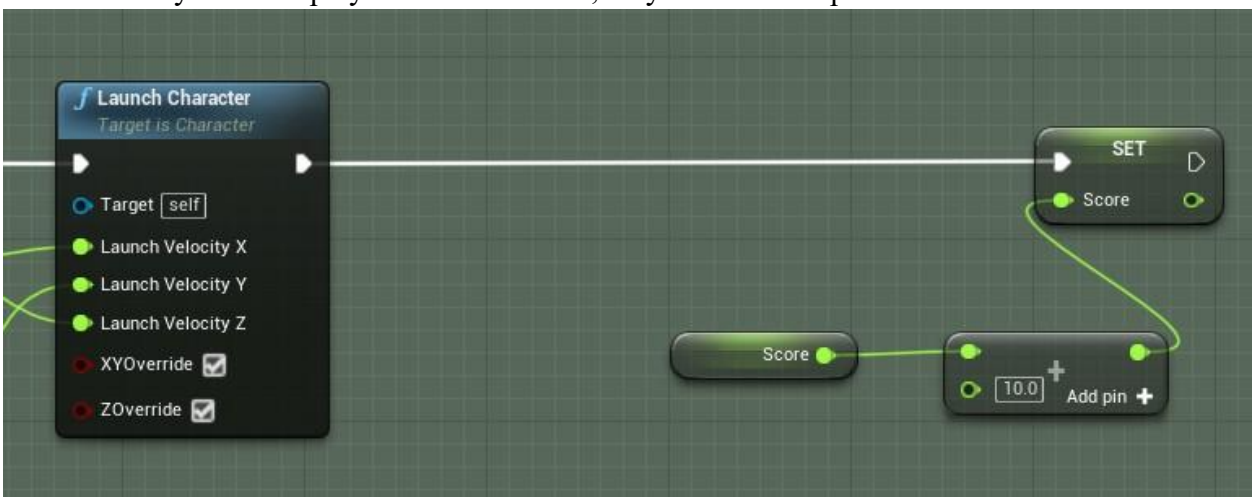
Frogger Styled Game Instructional Guide



- This variable represents the player's location when they first start a level and when the player dies, they will respawn here.

Player Score

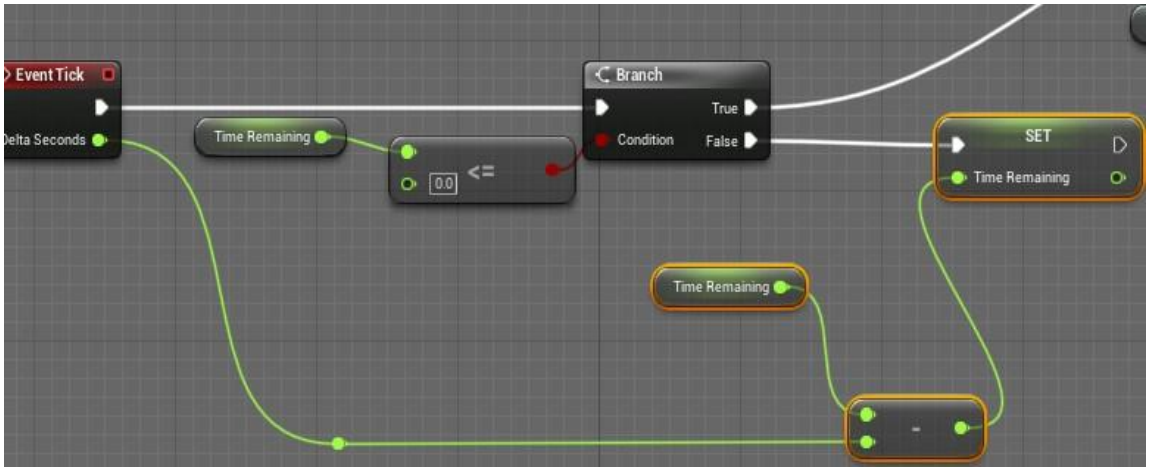
- Drag in the Score variable and connect to a float plus float node with a value of 10.
- Drag in the Score variable and make it a Set Score node.
- Connect the float plus float node to the Set Score node and then connect it to the launch character node for the W key input.
- With this every time the player moves forward, they will earn 10 points.



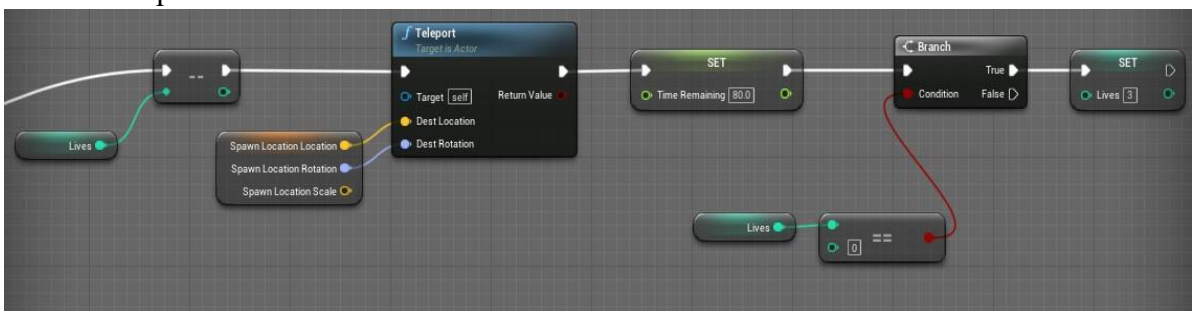
Player Timer

- Get a Branch node and connect it to the Event Tick node.
- Drag in the Time Remaining variable and connect it to the top part of a float <= node. Set the value to zero because we are checking if the player time has run out.
- Connect the bool pin of <= node to the branch node.
- If the player still has time left, then we are going to subtract the delta seconds on the Event Tick from the Time Remaining variable and set Time Remaining.
- Connect the set Time Remaining variable to the false pin of the branch node.

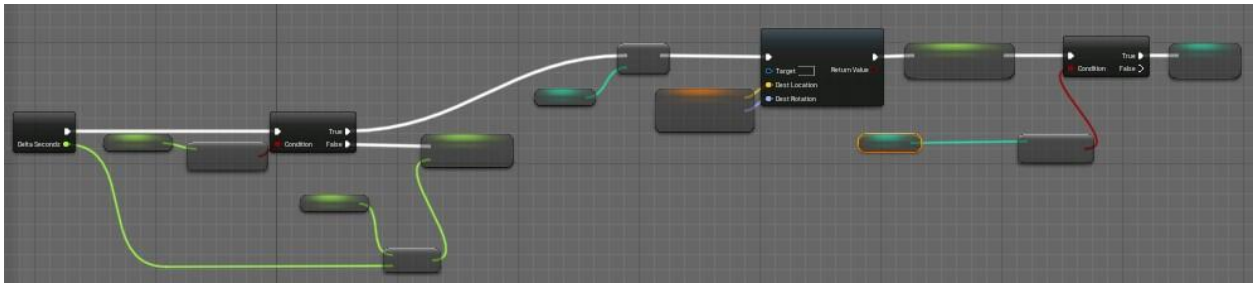
Frogger Styled Game Instructional Guide



- If the Time Remaining variable is less than or equal to 0 then we are going to remove a live from the player. Add in a decrement int node, connect the Lives variable to it, and then connect the node to the true pin of the branch node.
- Next, we are going to teleport the player to the spawn location in our level. Do this by adding in a Teleport node, drag in our Spawn Location variable, split the variables pin and connect the location and rotation to the teleport nodes corresponding pins.
- Next, we are going to set the timer back to its initial value to restart the timer.
- If the player ran out of lives because they ran out of time, then we are going to refill the players lives.
- We can do this by dragging in the Lives variable, check if Lives equals 0 with the equal node and connecting the bool pin from the equals node to a new branch node.
- If the Lives variable equals 0 then we are going to set the Lives variable to 3 and connect it to the true pin of the branch node.



- One we're done we will have a working timer system which we can show when we add UI to our game.

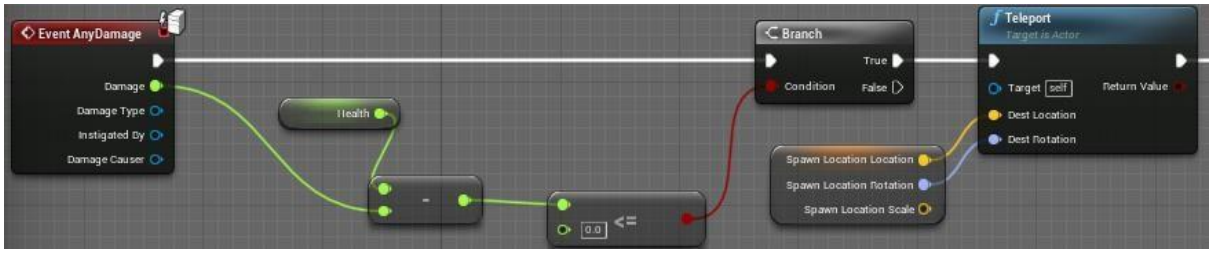


Frogger Styled Game Instructional Guide

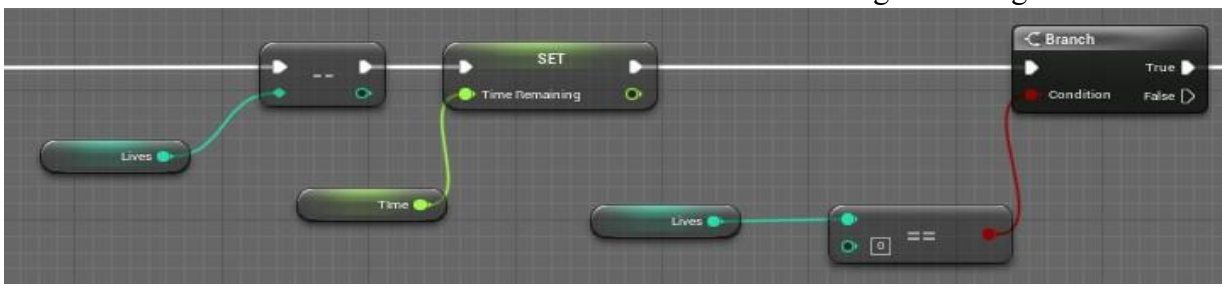
Player Health/Damage Handling

Now that we have health and live variables, we are now going to make a way for the player to react to damage from future obstacles and a way for the player to respawn.

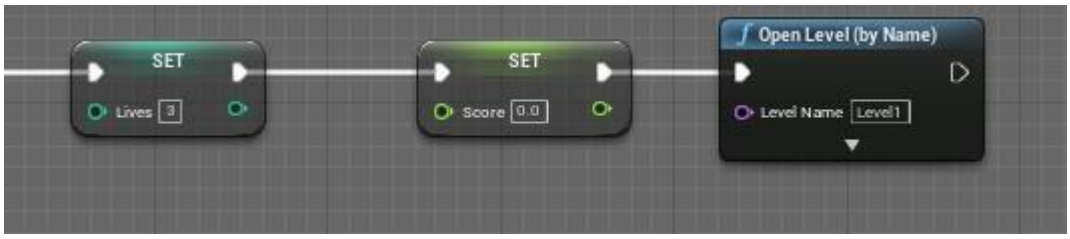
- Add in an Event AnyDamage node. Drag in the Health variable and drag off the pin and type in the minus symbol to get float – float.
- Connect the damage pin to the bottom of the subtraction node to subtract damage from the health variable.
- Drag off the return pin on the subtraction node and type <= to get the node. Get a branch node and connect the Boolean return pin of the <= to it.
- Get a Teleport node and connect it to the true pin of the branch node. Drag in our Spawn Transform split its pin and connect the Spawn Location and the Spawn Location Rotation to the corresponding pins on the Teleport node.



- To subtract lives when the player dies, we are going to use a decrement int node which you can get by right clicking and typing --. Connect the Lives variable to the decrement int node which will subtract one from the total each time the node is gone through.



- Just like we did for the player timer we are going to check if the Lives variable equals zero. If it does, then we are going set the Score variable to 0 and restart the game by opening the first level using an Open Level (by name) node. Make sure the level names match or the node will node work.



Frogger Styled Game Instructional Guide

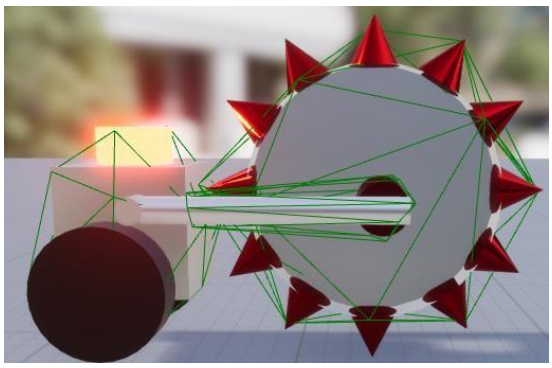
Part 4: Making Obstacles

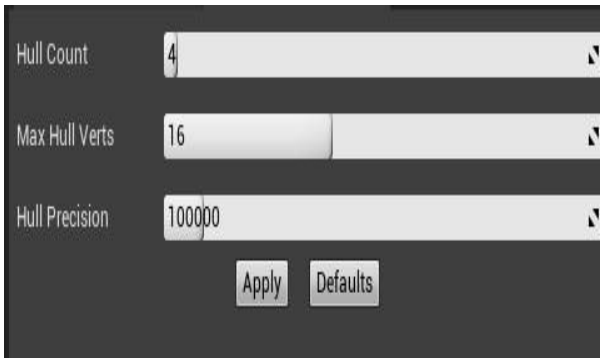
With the key player systems done and implemented, we can now make obstacles.

Making a Moving Obstacle

I'm going to show you how to make an obstacle that can move across the screen.

- Before I start coding, I'm going to make a static mesh for our first moving obstacle that I will call Saw Bot.
- Open this static mesh, click on collision, and then click on auto convex collision. This will make a tab called convex decomposition where we will click the apply button. This is just one option. If you want to you could use a different collision type or add the collision in the actor blueprint.

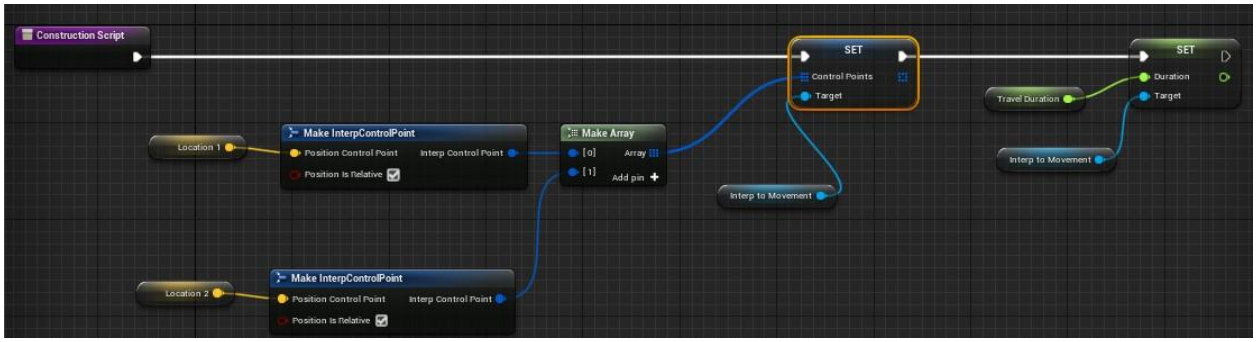




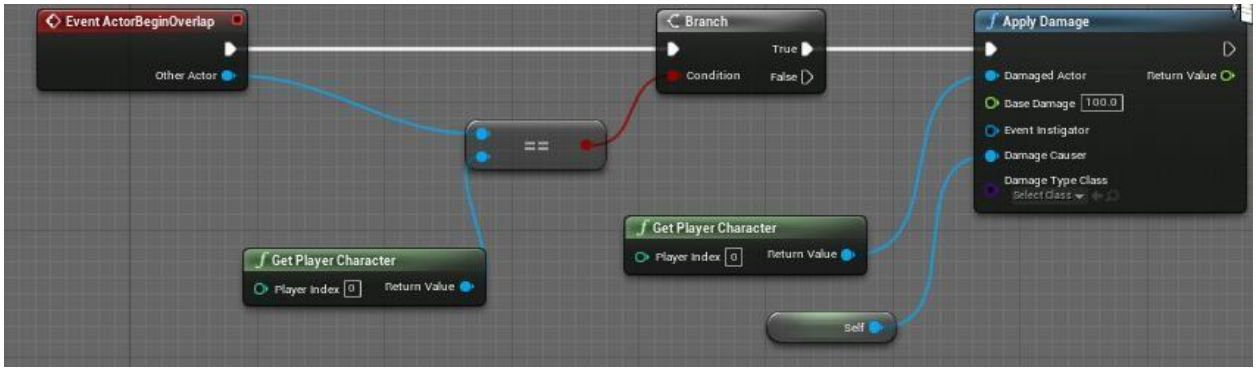
- Like with levels I recommend making a folder for obstacles.
- Make an actor blueprint class for the moving obstacle and name it whatever.
- For our obstacle we are going to make 2 vector variables called Location 1 and Location 2 and then we are going to make a float variable called Travel Duration.
- For the vector variables check the show 3d widget box which will also help with obstacle movement later.
- Leave all the values of these variables at 0 but click the eye to the right of them to make them instance editable. Making these variables instance editable will allow us to change the details for the actor for each of its instances in the level. This means that if we have multiple of the same obstacle, we can make each have different travel paths and speeds.

Frogger Styled Game Instructional Guide

- Add an Interp to movement component to the actor which will allow our obstacle to move. Change the interp to movement component's behavior to ping-pong so that it can go back and for.
- Go into the construction script and drag in both of the vector variables that we made.
- Drag of each of the vector variable pins and type Make InterpControlPoint to get the node.
- Add in a Make Array node, add a pin to the node, and connect the two Make InterpControlPoint to the make array node.
- Right click the Arry pin on the Make Array node and promote it to a variable. Drag in the Interp to Movement component and connect it to the Target of the array variable we just set.
- Drag off from the Interp to Movement component and set the Duration. For the duration value plug in our Travel Duration variable.

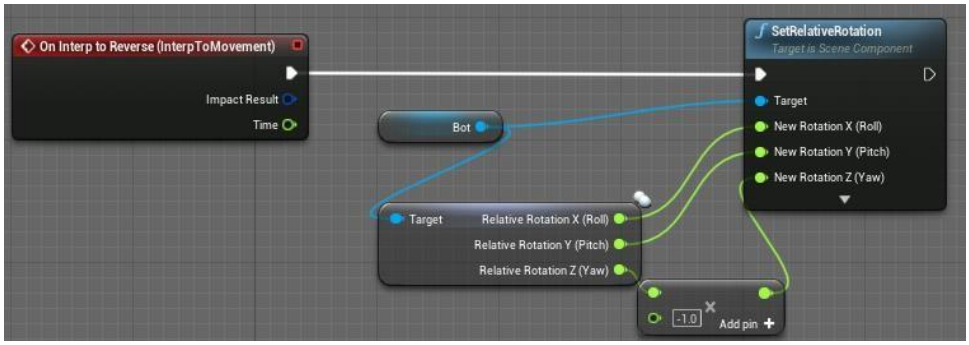


- To make this obstacle do damage go to the Event ActorBeginOverlp in the Event Graph. Drag of from the Other Actor pin get an == node.
- Add a Get Player Character node and connect it to the == node.
- Then add a Branch node and connect the bool pin of the == node to the condition pin oof the Branch node.
- If the actor that the obstacle overlapped was the player, then we will deal damage equal to the player's health. This can be done by using an Apply Damage node with the damaged actor being the player and the base damage being the number we set the player health to. This will make it so that the obstacle only damages our player character.



Frogger Styled Game Instructional Guide

- The next thing I'm going to do is make it so that our obstacle rotates when it reaches an end point. To do this right click the Interp movement component, go to add event and add the On Interp to Reverse event. This event will activate when the obstacle reaches one of its end points.
- Drag in the obstacle mesh in the Event Graph and drag off its return pin and get a SetRelativeRotation node.
- Get the relative rotation of the mesh and then split the pin to get the X, Y, and Z rotations.
- Plug the X and Y rotation into the SetRelativeRotation node but for the Z rotation multiply it by -1 and then plug it into the node.



- Now we can place our obstacle in the level and because it is instance editable, we can have multiple obstacles with different speeds and locations.
- As you can see, with can choose where our obstacle will go by either directly typing or moving the 3d widgets that represent our Location 1 and Location 2 vector variables. We can also choose how long the obstacle takes to move between one or two locations by using the travel duration variable.
- You can copy and paste this obstacle and put it in different spots in your level.

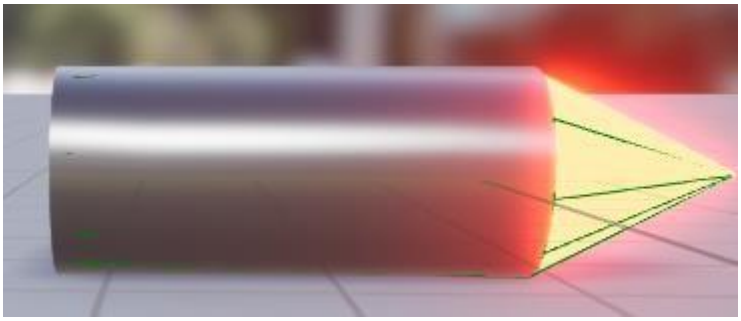
- If you want to make your moving obstacle, go vertically or diagonally, you must do is move Location 1 and Location 2 in the level. You can even duplicate your original moving obstacle blueprint and just change the mesh.

Frogger Styled Game Instructional Guide

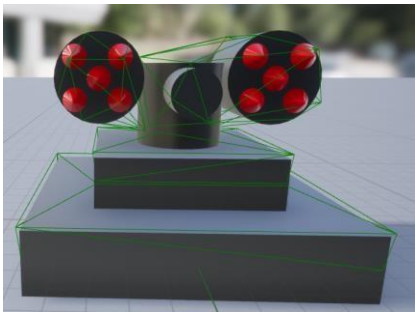
Making a Projectile Obstacle

Now that we have made a moving obstacle let's try something different and make an obstacle that fires projectiles.

- I will start by making a custom mesh for my projectile. Do it the same way we've done previously and make sure that it has collision.

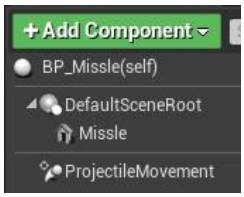


- I will also make a custom static mesh for my projectile launcher which will be look like a missile turret.

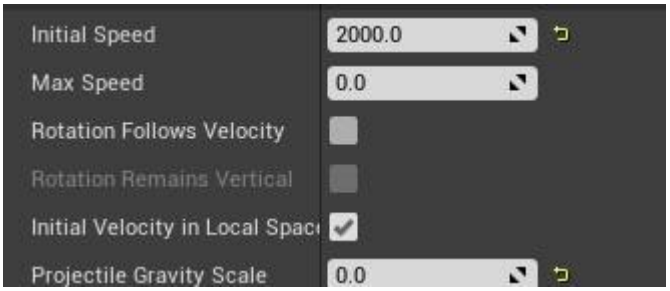


- First, we are going to add a projectile movement component to our missile which will make our missile move like a projectile.

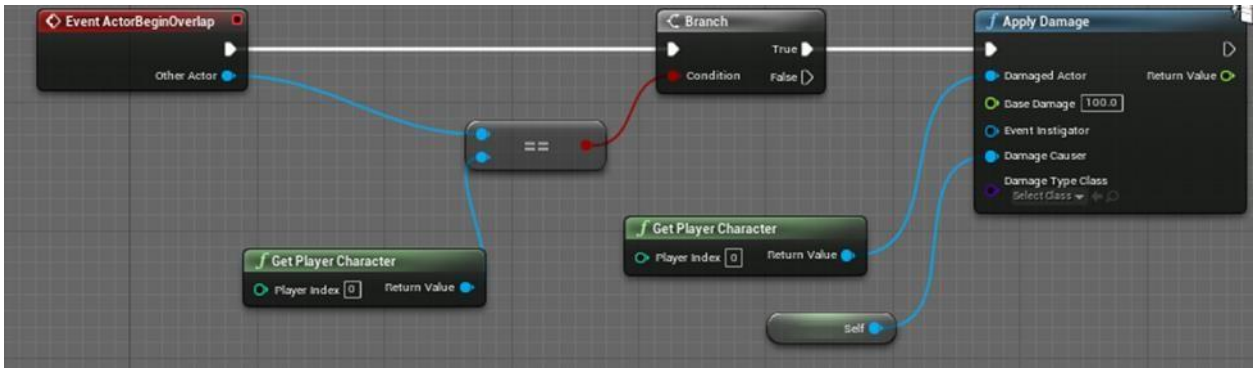
Frogger Styled Game Instructional Guide



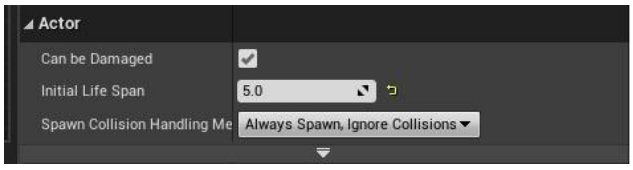
- Click on the Projectile Movement component and set its initial speed to 2000 and its projectile Gravity Scale to 0.



- Next make an actor blueprint for our projectile which I will call Missile. The way to do damage is the same way we did for our moving obstacle so you can just copy and paste that code here for our missile.

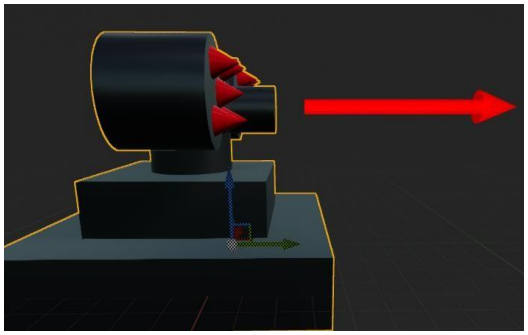


- Now we are going to click on Missile(self) right below the green add component button and then set the life spawn of our projectile to 5 so that it doesn't' stay in the level forever.

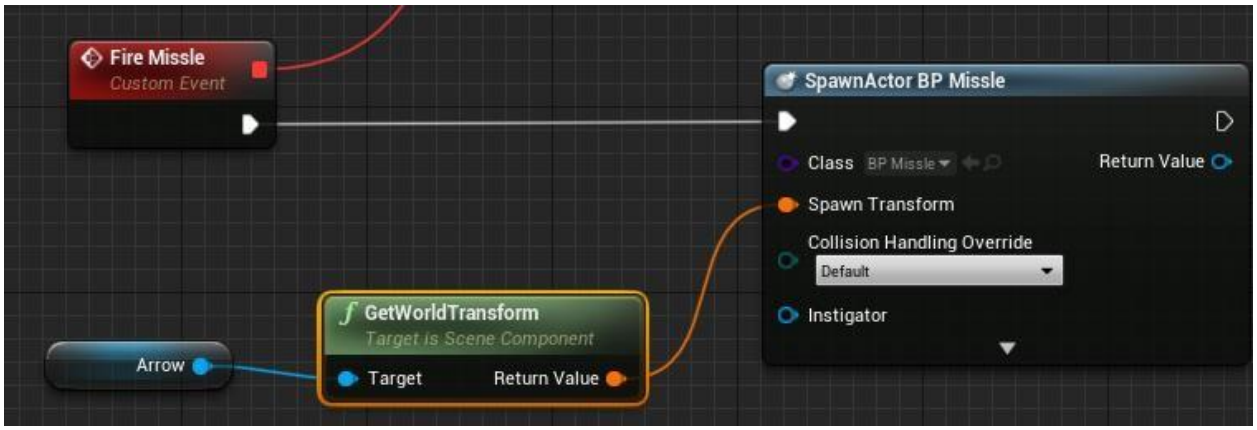


- Our projectile is ready to be fired by our turret.
- Next, we are going to make a blueprint actor for the turret and use our turret mesh for it.
- Add an arrow component to the actor and place it in front of the turret like this.

Frogger Styled Game Instructional Guide



- Next, we are going to go into the event graph and make a custom event which I will call Fire Missile.
- From this custom event we are going to attach a spawn actor from class node and for the class we are going to choose our projectile actor blueprint.
- Drag in the arrow component and get its world transform which will attach to the Spawn Actor node's spawn transform.



- Next drag off from the output delegate on the custom event and type set timer by event and connect it to Event BeginPlay.
- Promote Time to a variable that's instance editable and check the Looping box so that the turret will constantly fire.

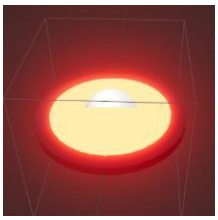
- Now you can place multiple turrets in our level with different firing times.

Frogger Styled Game Instructional Guide

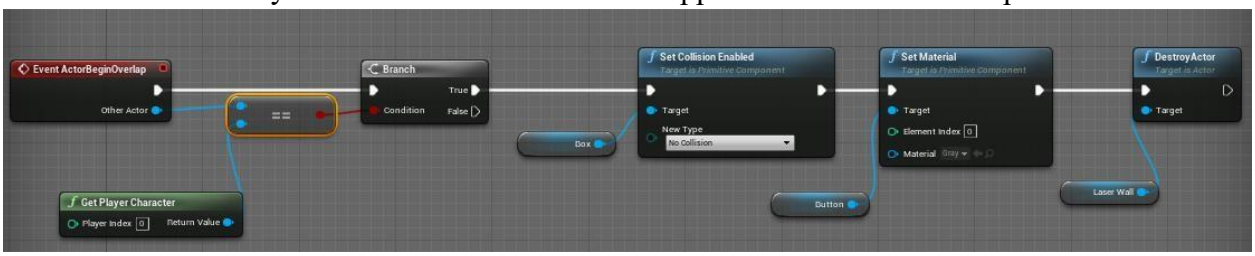
Making a Button Controlled Obstacle

Next, we are going to make an obstacle that can be turned off by pressing a button or pressure plate.

- First, we must make the obstacle that will be affected by the button. Since I want a futuristic theme, I will make the obstacle a laser that kills the player when they overlap it.
- Next, we are going to make a button actor which I will call BP_LaserButton. I will add in a button mesh I made along with a box collision component.



- Go to the Event ActorBeginOverlap and for my purposes I'm going to make it so that the laser actor is destroyed, and the button light will change color to signify its destruction. To do this we first have to make an object reference variable for our obstacle that is connected to our button and make it instance editable.
- Then we are going to check if the other actor equals the player character and if it does drag in the box collision component and set its collision to No Collison using the Set Collision Enabled node. This will make it so that our button only works once.
- Next, I'm going to set the element 0 of my button mesh to grey.
- Finally, we are going to drag in our obstacle variable which for me is a laser wall that is connected to a Destroy Actor Node so that it will disappear when the button is pressed.



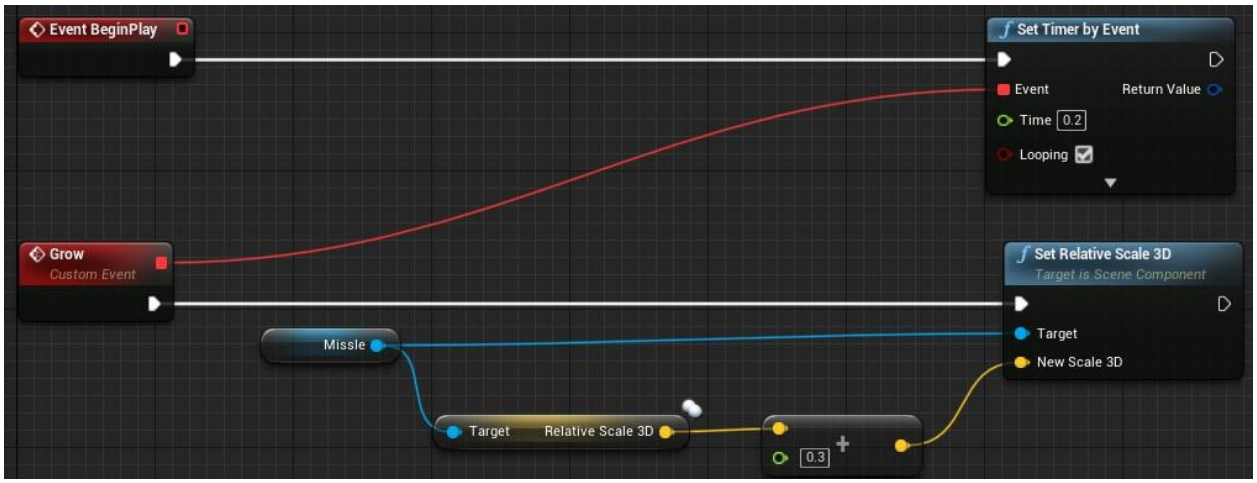
- When you place the button and obstacle into your levels you can go to the details tab of the button and choose an obstacle to attach to it.

Making an Obstacle that Grows

Next, we are going to make an obstacle that can grow overtime.

Frogger Styled Game Instructional Guide

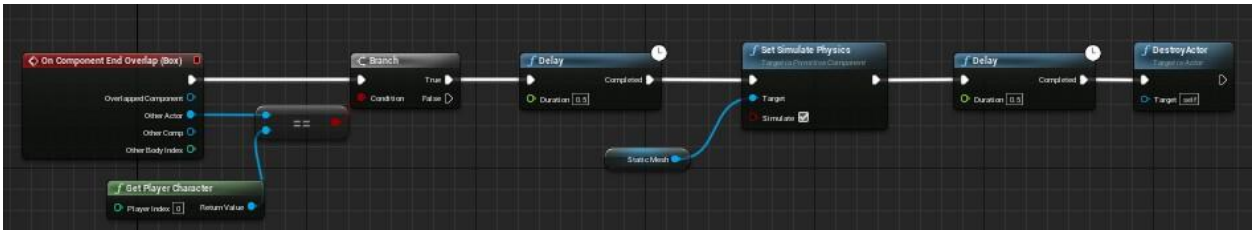
- I'm going to make my growing obstacle using a laser ring projectile. If you want to make a ring projectile, then use the Shape_Torus static mesh.
- Use the same damage code as the previous obstacles.
- Next, we are going to make a custom event called Grow.
- Drag in the mesh component and get its Relative Scale 3D vector Variable. Connect this vector variable to a vector plus float node and set the float to whatever works for you which for me is 0.3.
- Drag off from the mesh component get a Set Relative Scale 3D node and then connect the vector plus float node's return value to the Set Relative Scale 3D's New Scale 3D pin.
- Next drag from the output delegate and get a Set Timer by Event node. Set Looping to true and I'm going to set Time to 0.2
- Connect the Set Timer by Event to the Event BeginPlay. Since this I'm making a projectile, I will set the actor life span to 8.



Making a Falling Platform

- For my platform I will use a cube static mesh along with a box collision.
- Right click the box collision component and add a On Component End Overlap event. You could use a Begin Overlap event instead, but I found that End Overlap worked best for me.
- For the end overlap event, check if it's the player and if it is we are going to use a delay node with a duration of 0. 5
- Next, we are going to drag in the static mesh component and connect it to a Set Simulate Physics node with the Simulate bool box checked.
- After that we are going to connect it to another Delay node with a value of 0.5 and at the end of that a DestroyActor node.
- This will make it so that once the player moves off the falling platform it will drop after 0.5 seconds and will disappear after 0.5 seconds.

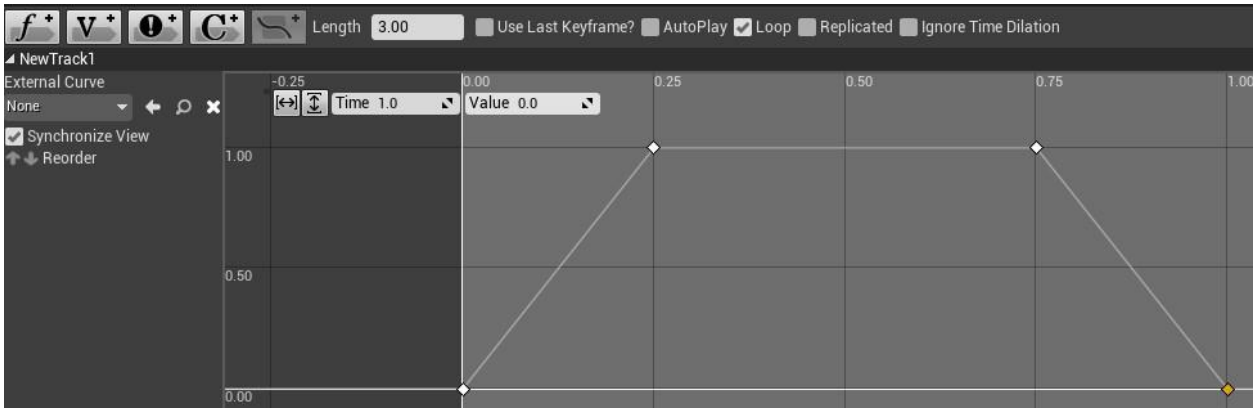
Frogger Styled Game Instructional Guide



Making a Spike Obstacle

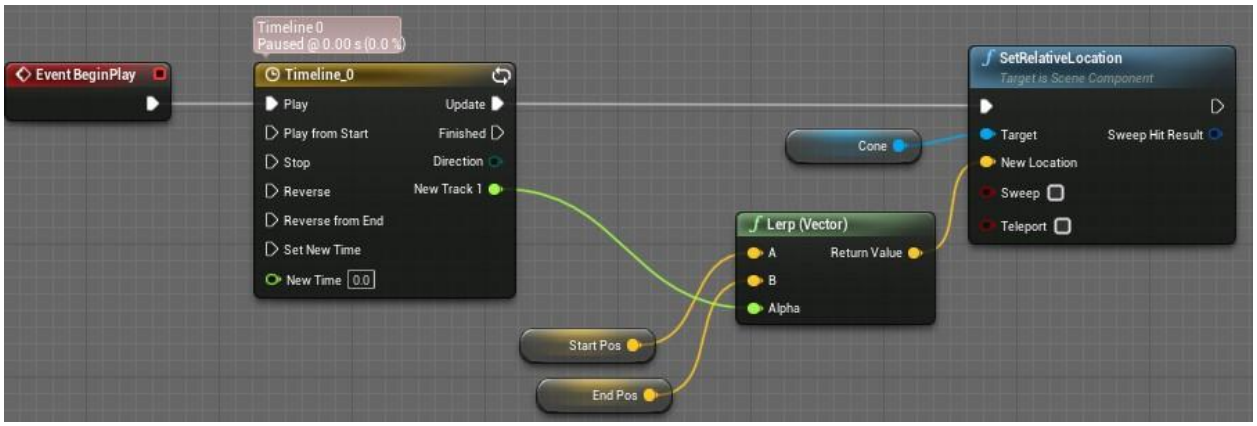
Now we are going to make a spike obstacle. To and do this we can either use the same code we did for the moving obstacle and line up the location variables vertically or use a timeline. For the spike obstacle I will use a timeline to make it move.

- First, I will make an actor blueprint called BP_Spike.
- Simply add in a cone mesh to make our spike.
- Then add in a timeline node and attach it to begin play. Double click the node to open it in a float track by clicking this button. 
- Set the length 3 which represents seconds and right click to and a key to the curve float. Add one key with a time of 0 and a value 0, a second key with a time of 0.25 and a value of 1, a third key with a time of 0.75 and a value of 1, and a fourth key with a time of 1 and a value of 0.

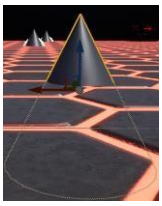


- Go back it the event graph and make two vector variables called Start Pos and End Pos. For Start Pos set the Z value to -50 and for End Pos set the Z value to 50. These variables will determine how far the spike will go up and down.
- From the float track pin drag and add in a Lerp (Vector) node and connect the Start Pos to the A pin and the End Pos to the B pin.
- Drag the cone into the event graph and connect it to a SetRelativeLocation node. Connect the return value of the Lerp (Vector) node to the SetRelativeLocation node.
- Next, add the damage code the same way we did for the previous obstacles and spike obstacle is now done.

Frogger Styled Game Instructional Guide



- Now you can add multiple spikes in the level and adjust it in the manner that you desire. I also recommend placing the spike slightly in the ground.



Part 5: Making the Player Objective

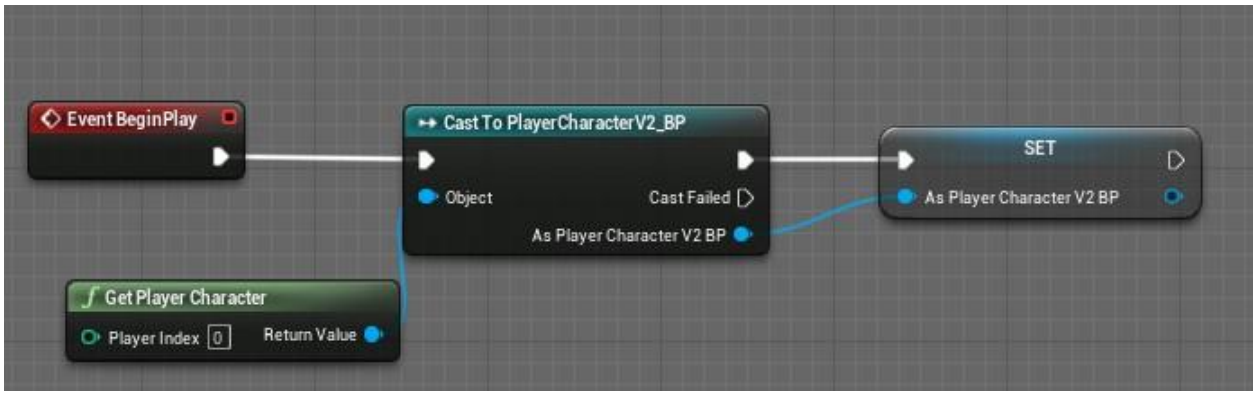
Now that we have completed some basic obstacles we will move on to making a player objective. This objective will be our version of the Frogger Homes in Frogger

Making a Frogger Home

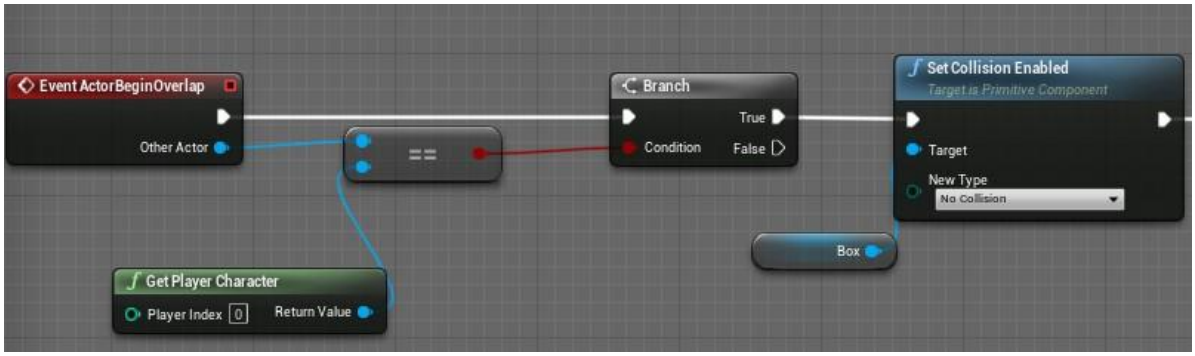
- First, I'm going to make a static mesh for our objective.
- Next, make a blueprint actor which I'm going to call BP_Home.
- We will add in our objective static mesh which I called home pad and add in a box collision and a static mesh or skeletal mesh depending on which way you decided to make your character from the previous instructions.
- If you use a skeletal mesh, then add in the mannequin you used for the player and check the hidden in game box to make it invisible when we play.
- If you use a static mesh, then you can do the same thing as the skeletal mesh or you can use a set static mesh node which I will go over later.
- For this example, I have both the skeletal and static mesh, but you should only choose one.

Frogger Styled Game Instructional Guide

- Go to Event BeginPlay and connect a cast to the player character to the Event Begin Play node with the object of the cast connected to a get player character node. Then promote the As Player Character to a variable which we will use later.

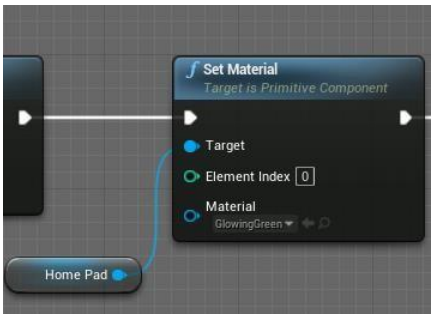


- Go to Event Begin Overlap and check to see if it was the player character that overlapped the actor like we've done previously.
- If it was the player, then add in a Set Collision Enabled node for the box collision and set the new collision type to No Collison so that the player can reactivate the same objective multiple times.

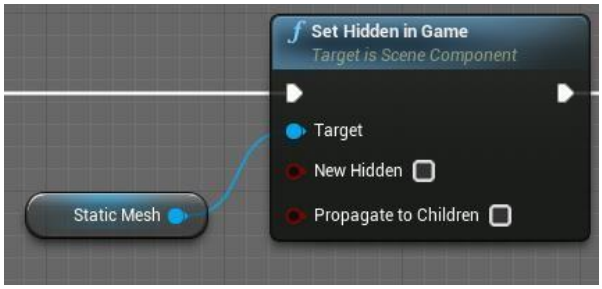


- For visual feedback I made it so that the red light on my home pad changes to green when the player steps on it. This can be done with a Set Material node that's targets Element Index 0 which is the home pad's light and by changing it to a green light.

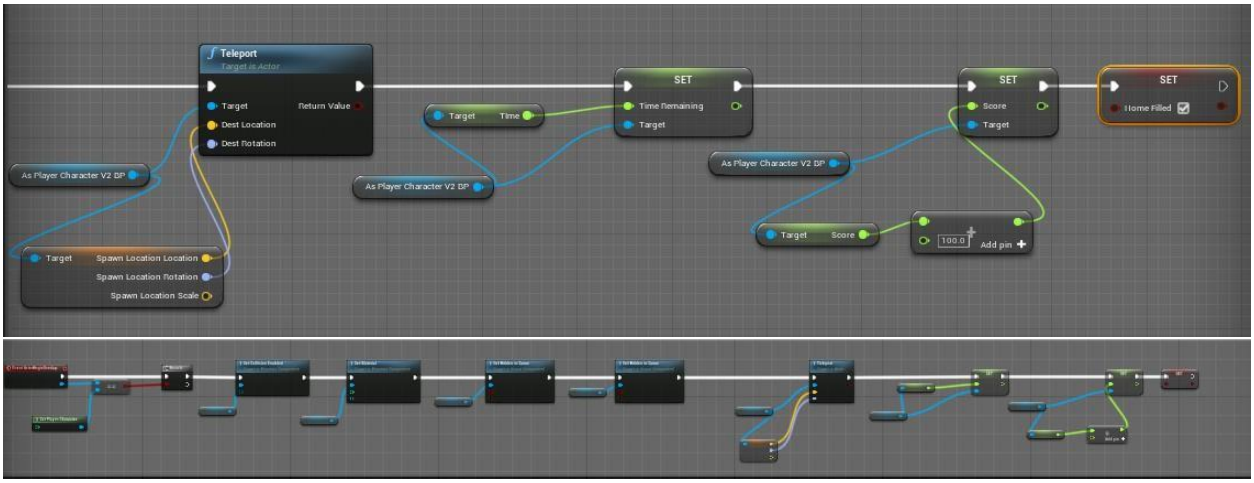
Frogger Styled Game Instructional Guide



- After that we are going to set the static mesh hidden in game to false with a Set Hidden in Game node.



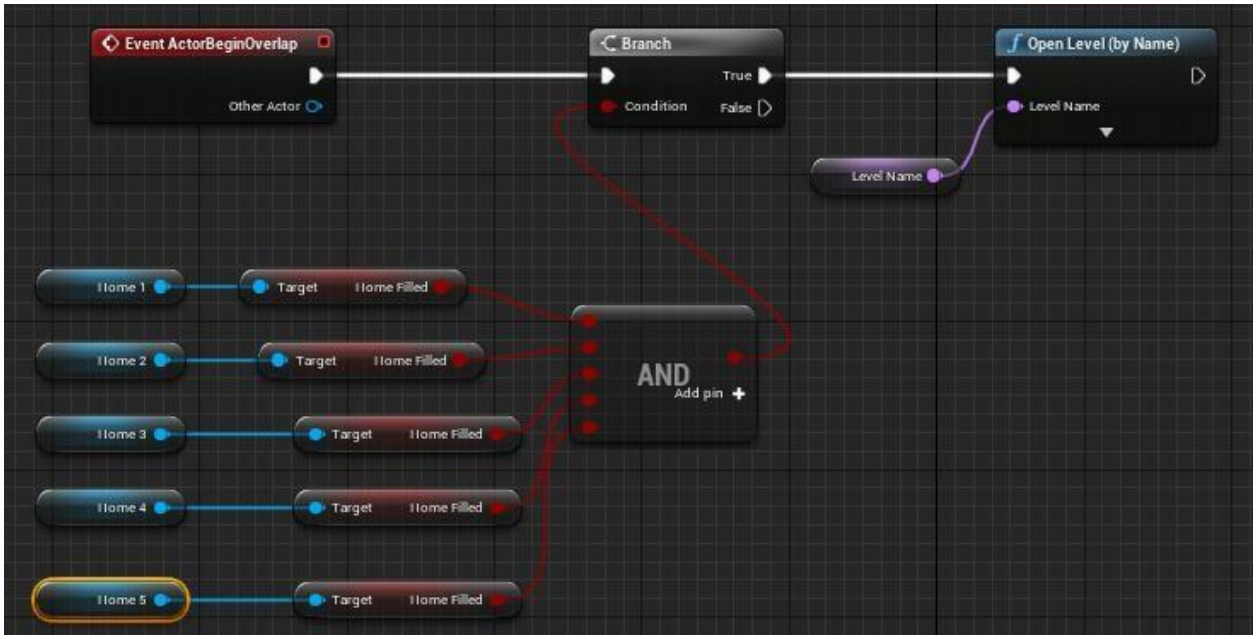
- Next, we are going to teleport the player back to the start location of the level. To do this add in a teleport node and make the target our As Player Character variable that we made at begin player and from the As Player Character variable get the Spawn Transform, split the pin, and plug in the location and rotation into the Teleport node.
- After that use the player character variable to get the Time Variable and use the Time variable to set the Player's Time Remaining variable which will refill our player time.
- Then we are going to get the score variable from the player character variable, add use a float plus float node to add 100 to it, and then set the score variable.
- Next, we are going to make a bool variable called Home Filled and set to true at the end of our code.



- Next, we are going to make a way for the game to track how many homes have been filled and when to move to the next level.

Frogger Styled Game Instructional Guide

- To do this make an actor blueprint which I will call NextLevelCondition.
- Add in a box collision component.
- Next, since Frogger has five frogger homes for the player to reach, we will have five homes as well. Add in a variable called home which will be an object reference to our BP_Home objective actor. To get an object reference when you're choosing a variable type our objectives name in the search bar and when you hover over it and there will be an option for object reference.
- Make five of these home variables and make them all instance editable then make an instance editable variable called Level Name.



- Drag out the 5 Home variables and get the Home Filled bool for each of them.
  Then add in a AND bool node and add enough pins for each of the Home Variables.
  Connect the Home Filled Bools to the AND node and then connect the return pin to a branch node.

- Connect the Branch node to the Event ActorBeginOverlap node and from the true pin connect an Open Level (by name) node to with the level name being our Level Name variable.
- Go into the level and place 5 objective points or Home Pads for this example.
- Next put in the NextLevelCondition actor in the level and make it big enough to cover the entire level.
- In the details tab change the default values to the objectives we placed in the level using the drop-down menu and type the exact name of the level you want to go to which for me is Level2. Once everything is done, play and see that once all 5 homes are filled the player will move on to the next level.

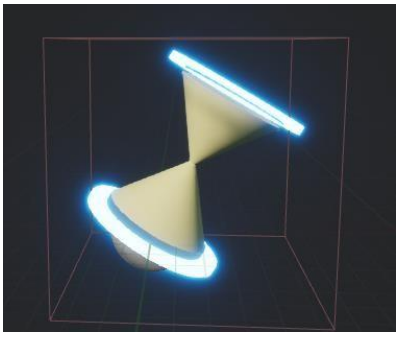
Frogger Styled Game Instructional Guide

Part 6: Making Player Pickups

Next, I will show you have to make pickups for our player. For this section I will show you how to make a time increase pickup. The code for it is simple and can be adjusted to affect aspects of our player and game.

Making a Time Pickup

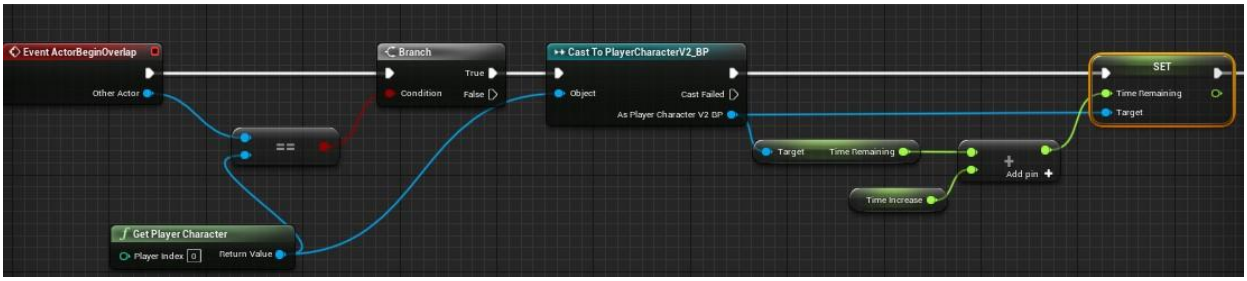
- For my time pickup I'm going to use a static mesh I made along with a box collision and rotating movement component.



- Next, we are going to get to the Event ActorBeginOverlap, check if it's the player, and then we are going to use the other actor pin to cast to our player character to get the Time Remaining variable.

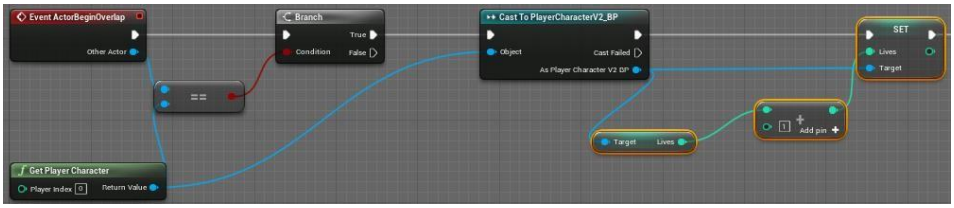
Frogger Styled Game Instructional Guide

- I'm going to add 10 seconds to the Time Remaining variable using a float plus float node.
- Use the as Player Character pin to set the Time Reaming variable and plug in the return value of the addition not to the Set node.



Making a Stat Increase Pickup (Lives, Health, etc.)

- If you want to increase the players lives, health, or jumping distance then the process will be the same, just change which variable you are getting from the cast node.



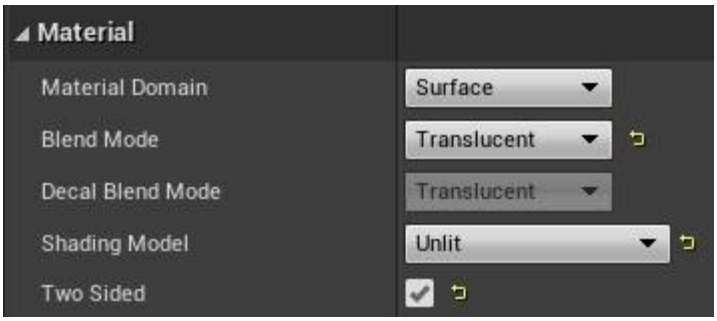
- If want to use text as a visual for your pick-ups add in a Text Render component. You will be able to change the size and color of the text to suit your needs.

Making a Shield Pickup

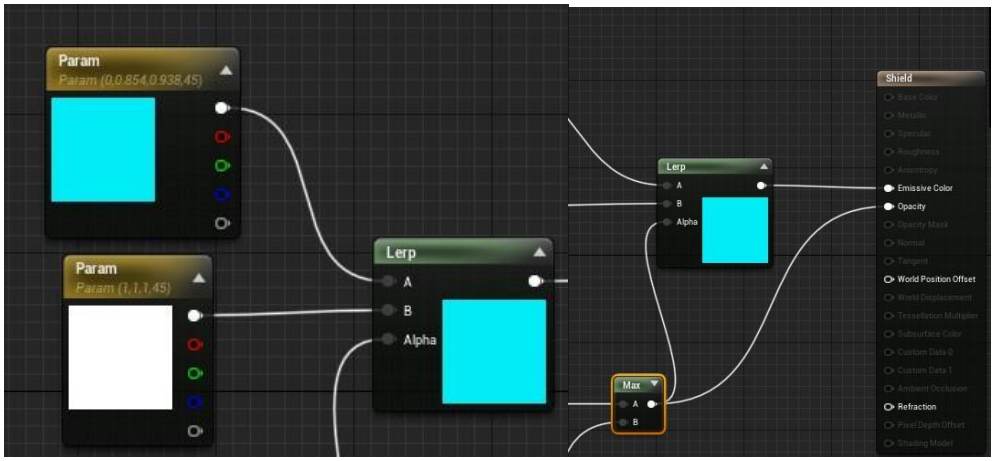
Next I will show you a way to make a pickup that will act as a shield that can prevent the player from getting damaged. Since I'm going for a more futuristic theme, I'm going to make this shield a bubble shield that covers the player.

- First, go to the details tab and under the Material section change the Blend Mode to Translucent, the Shading Model to Unlit, and check the Two-Sided box.

Frogger Styled Game Instructional Guide



- Add two Vector Parameter nodes with one being white and one being a color of your choice. I selected choose light blue.
- Connect the colored Vector Parameter node to the A pin of the Linear Interpolate/ Lerp node and connect the other Vector Parameter node to the B pin.
- Connect the output pin of the Lerp node to the Emissive Color pin of the Material Result node.

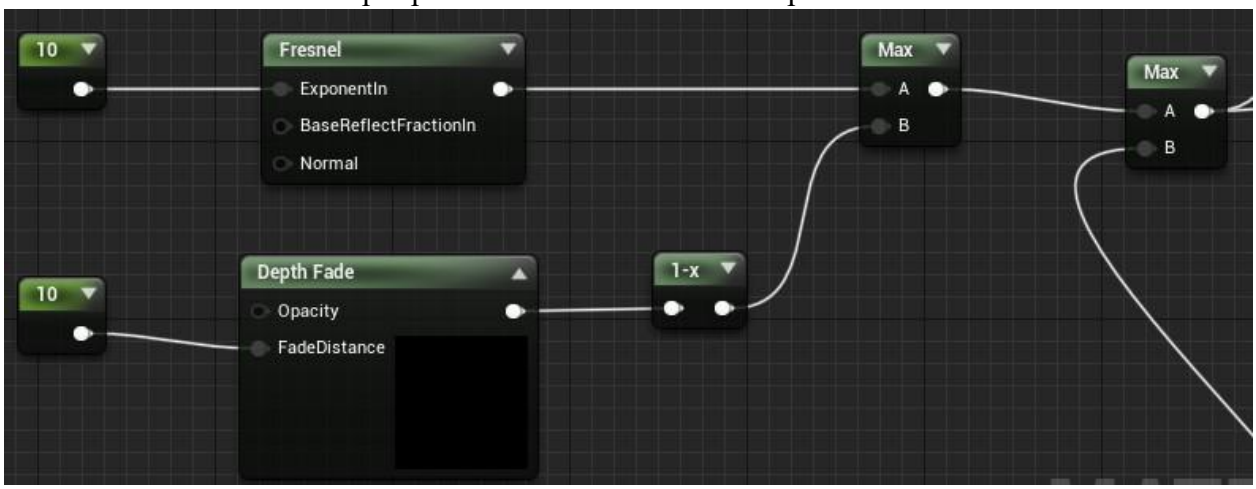


- Next make two Constant nodes and set their values to 10.

Frogger Styled Game Instructional Guide

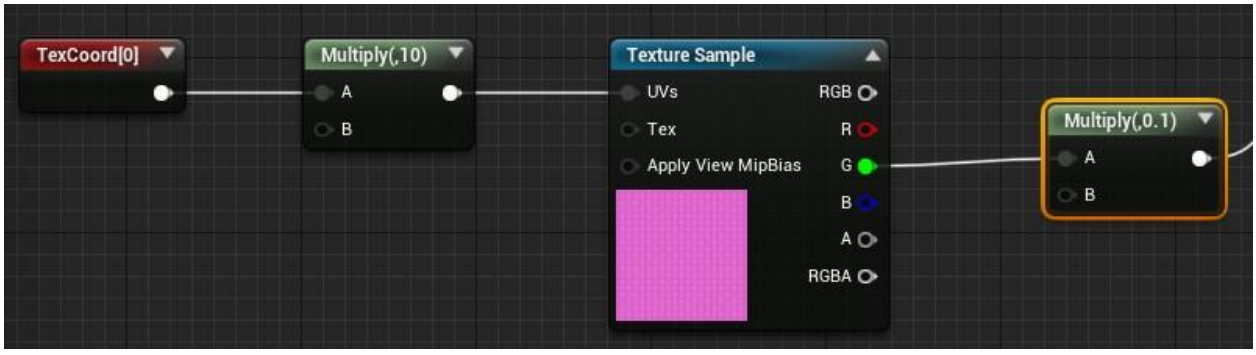
Connect one Constant node to the ExponentIn pin of a Fresnel node and connect the Constant node to the FadeDistance pin of a Depth Fade node.

- Add in a Max node and connect the output pin of the Fresnel to its A pin.
- Then connect a OneMinus node to the Depth Fade's output pin.
- Connect the OneMinus node's output pin to the Max node's B pin.
- Connect the Max node's output pin to another Max node's A pin.



- Next add in a TextureCoordinate node,
- Connect a Multiply node to the TextureCoordinate node. Click on the Multiply node and set the Const B value to 10.

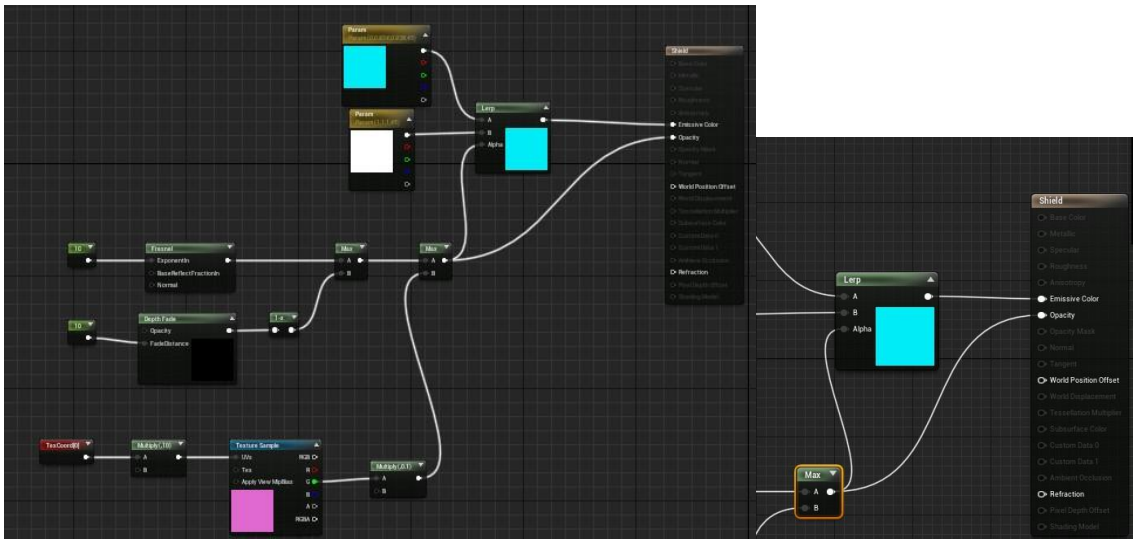
- Now we are going to add in a Texture Sample node where I will copy and paste from the M_Tech_Hex_Tile found in the Materials folder in the Starter Content folder. Connect the output pin of the Multiply node to the UV's pin of the Texture Sample node.
- Next Connect the Green G pin of the Texture Sample node to another Multiply node's A bin with the B value set to 0.1.



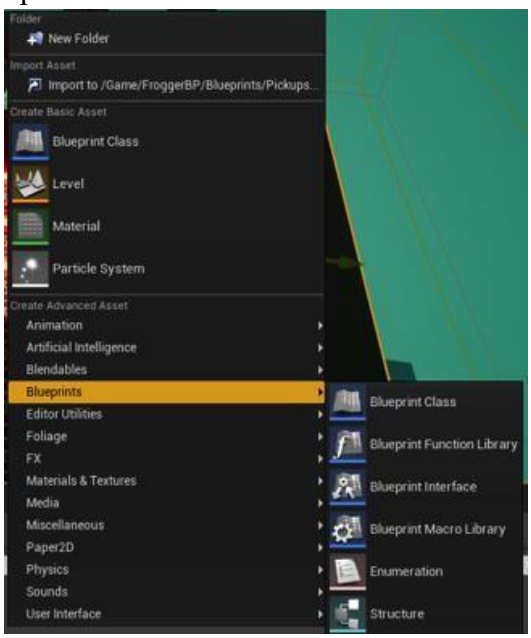
- Connect the output pin of the Multiply node to the B pin of the Max node we made earlier.

Frogger Styled Game Instructional Guide

-
- Connect the A pin of the earlier Max nodes to the Lerp nod's Alpha Pin and to the Opacity pin of the Material's return node.



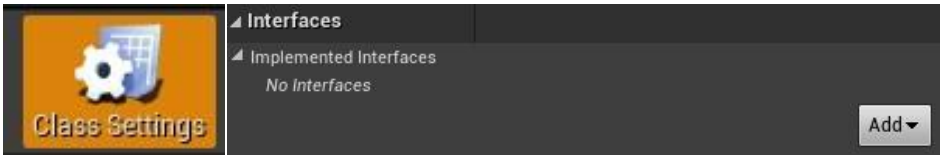
- Now we have shield material that looks like a bubble shield that you would see in a game like Halo.
- Before we start making the shield actor, we are going to make blueprint interface for our obstacles that I will call I_Obstacle. To make an interface similar to how we made our levels and other blueprints there will be a Blueprints section with a Blueprint Interface option.



- Go into all of the obstacles or anything that damages the player and in the blueprint editor there is a button called Class Settings. Click on it and under the details section there will be a section labeled Interfaces with an add button. Click the add button and

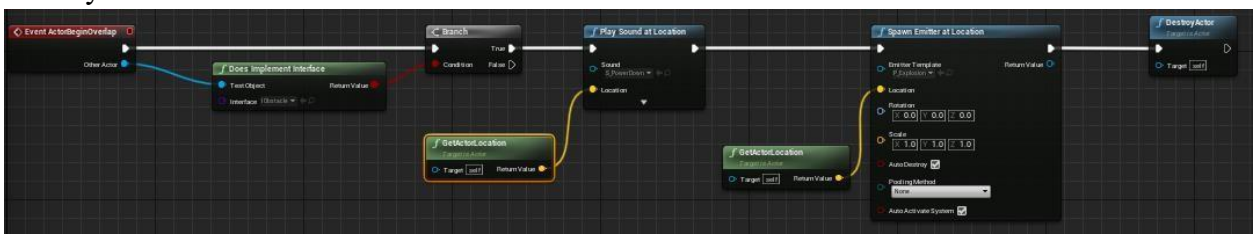
Frogger Styled Game Instructional Guide

scroll until you see the interface we made. Click on it and it will now be added to the blueprint.



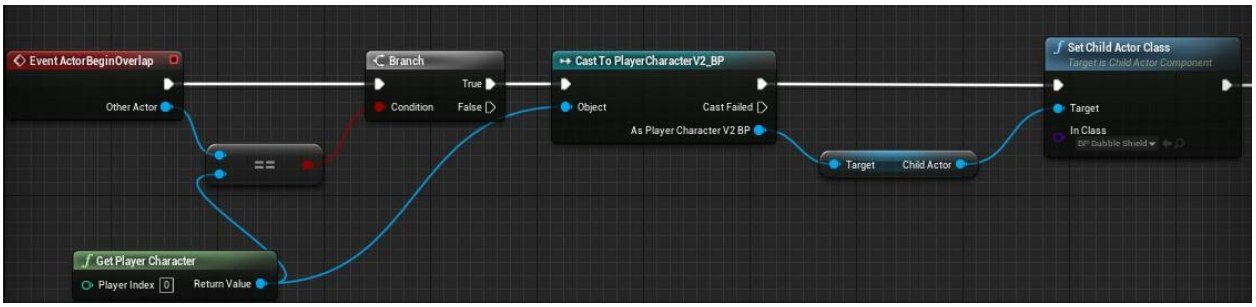
Next go into the player character blueprint and add a Child Actor component with the Child Actor Class set to none. Make sure the Child Actor component is on the player character.

- Next, we are going to make the actor for the shield. Make a new actor blueprint which I will call BP_BubbleShield.
- Open the actor blueprint and add in a sphere collision component with the collision set it to overlap all.
- Drag the sphere collision onto the root component to replace it.
- Add in a sphere static mesh and drag it on the sphere collision to attach it. Set the meshes collision to overlap all as well.
- Go to the Event ActorBeginOverlap and connect the Other Actor Pin to a Does Implement Interface node with the interface being the interface we made for the obstacles.
- Connect the Implement Interface node's Return Value to a Branch node.
- If whatever overlapped the shield has the I_Obstacle interface, then from the True pin I'm going to add a play sound node and spawn emitter node to show that the shield is gone with the location being the actor location. Then I will to destroy the shield with a Destroy Actor node.



- Now that the shield is done, we can now make the pickup.
- Make a new blueprint which I will call BP_Shield_Pickup.
- Make it the same as the other pickups but drag off of the As Player blue pin and get the Child Actor. Drag of the Child Actor pin and get a Set Child Actor Class node with the in class being our shield actor.
- Now when you step on the shield pick up a shield will form around the player that can block damage once.

Frogger Styled Game Instructional Guide

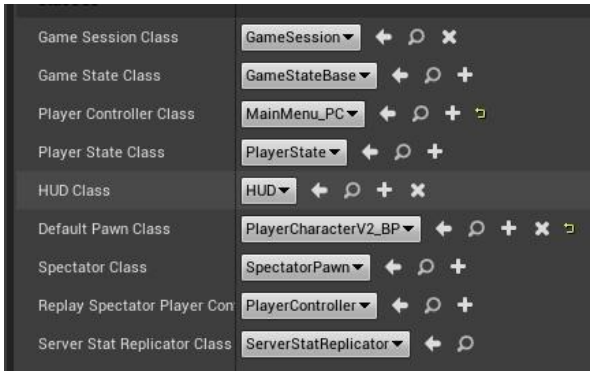
- 

Part 7: Making the Main Menu and UI

Now that we have the core systems and obstacles completed, it is time to make the main menu and UI for the player.

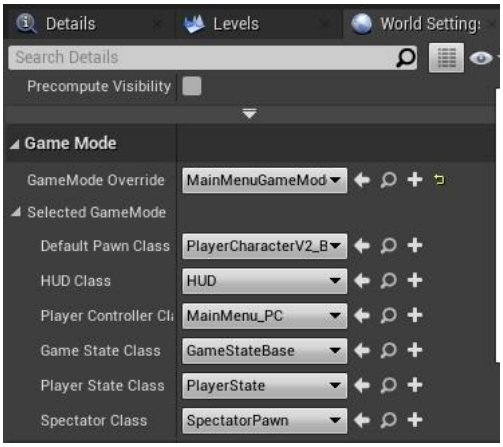
Making the Main Menu

- First, we are going to make a new map which I will call MainMenu_Map. This map will only hold our main menu, so you don't have to add anything like lighting to it.
- Next, we are going to make a new player controller and game mode blueprint which I will call MainMenu_PC and MainMenu_Gamemode.
- Go into the main menu player controller and check the show cursor box then go into the main menu gamemode and set the player controller class to the controller we just made and set the default pawn class to our player character.



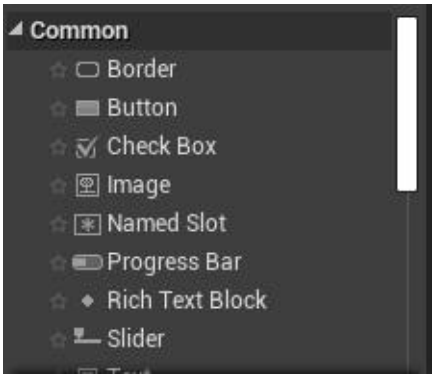
- Next go into the main menu world settings and change the game mode override to our main menu game mode, the default pawn class to none, and the player controller to our main menu controller.

Frogger Styled Game Instructional Guide



Next, we are going to create a widget blueprint for our main menu. For the main menu we are going to need a start and quit button. I also have a credits button but that's not necessary for our game.

- Under the Common section there will be a button along with other important elements and to add it just drag and drop it on to the canvas panel.



- Next, we are going to use text blocks to add names to the buttons. To do this simply drag a text block onto the button. Resize, position, and edit the text and buttons to whatever you think looks good and then we can move on to making the buttons work.

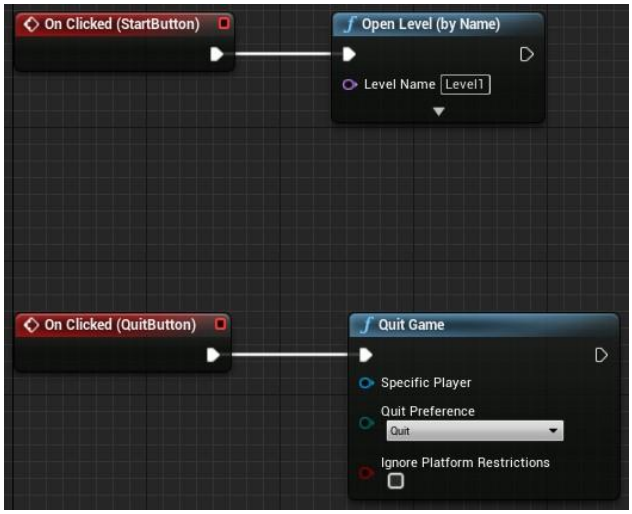
- On the top right of the screen there will be a tab called Graph which is used like an event graph. Click on it and you will see our buttons under the variable section.

Frogger Styled Game Instructional Guide

- 

- When you click on the button variable you will see that will be multiple green event buttons for the variable. For our purposes we are going to the On Clicked event for our Start and Quit buttons.

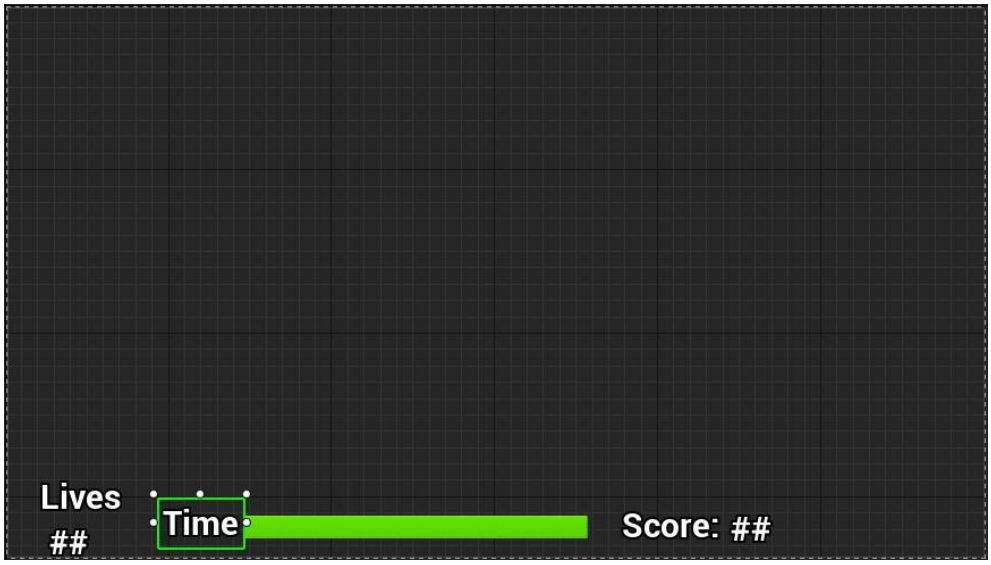
- For the start button's click open the first playable level using an Open Level node which is Level1 for me and for the quit button's on click use a Quit Game Node.



- With the main menu completed we will go into our project settings under the Maps and Modes section and set the Game Default Map to the main menu map.

- Make a widget blueprint for the player UI. Add in text blocks for the lives, score, and time. We will use a progress bar for time just like in Frogger. We are going to have a current life and current score text that will be used to show our lives remaining and player score.

Frogger Styled Game Instructional Guide

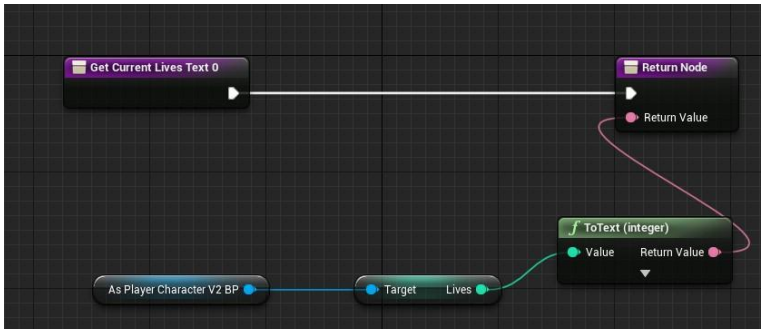


- First go into the graph and on the event, construct make a cast to the player character and make the return pin into a variable.

- Then, go back into designer and make the current lives text a variable by checking the Is Variable box.

- There will be a button called Bind click that and then click create binding which will allow us to bind our text to a variable.

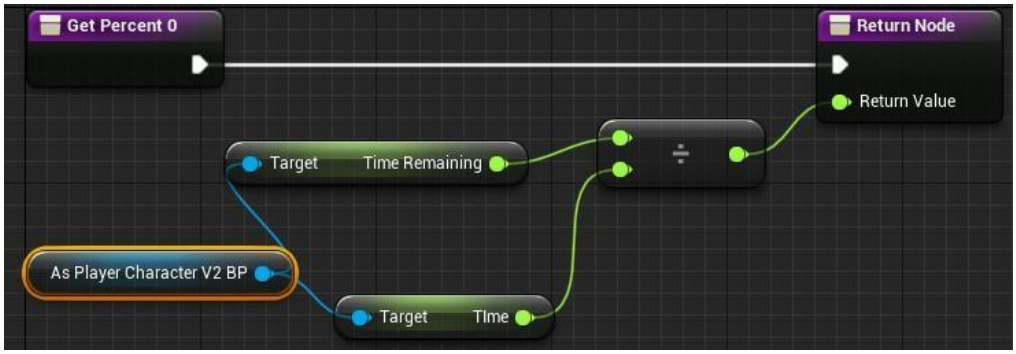
- Drag in our player character from the cast that we did and get the Lives variable. Plug this variable into the return value pin on the return node and it will convert our Lives variable into text that we will be able to see on screen.



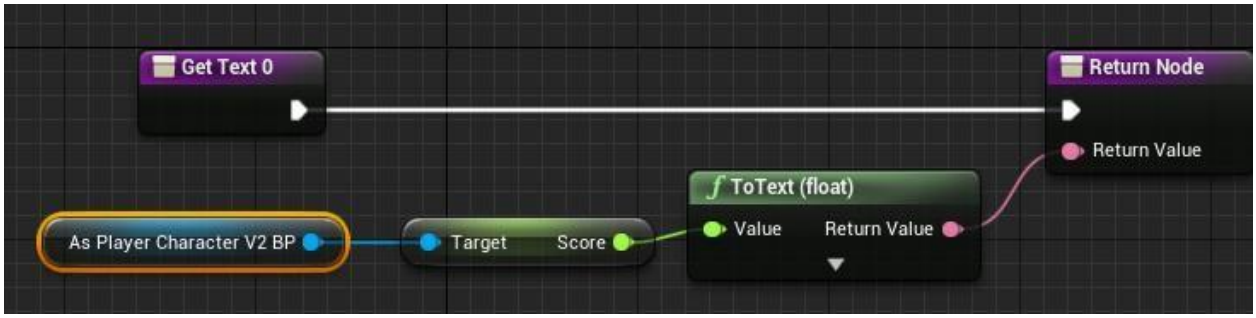
Frogger Styled Game Instructional Guide

- Go to the progress bar and set the percent to 1.0 and create a binding for it as well.

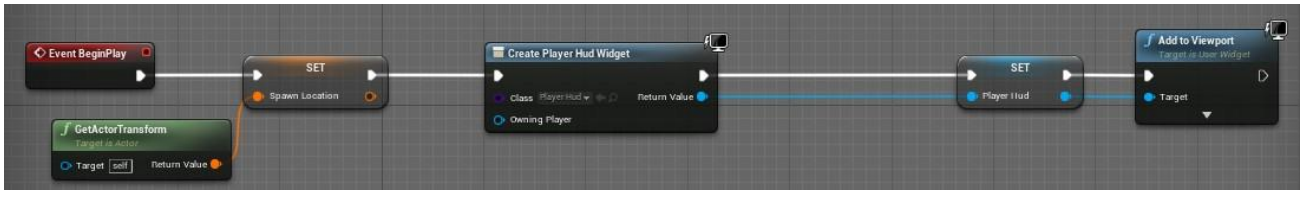
- Drag in the player variable and get the Time Remaining and Time variable.
- Divide Time Remaining by the Time Variable and plug it into the return node's return value pin. This will cause the bar to deplete when the player time goes down.



- Finally, go to the current score text and create a binding for it.
- Drag in the player variable and simply get the text and plug it into the return value on the return node.

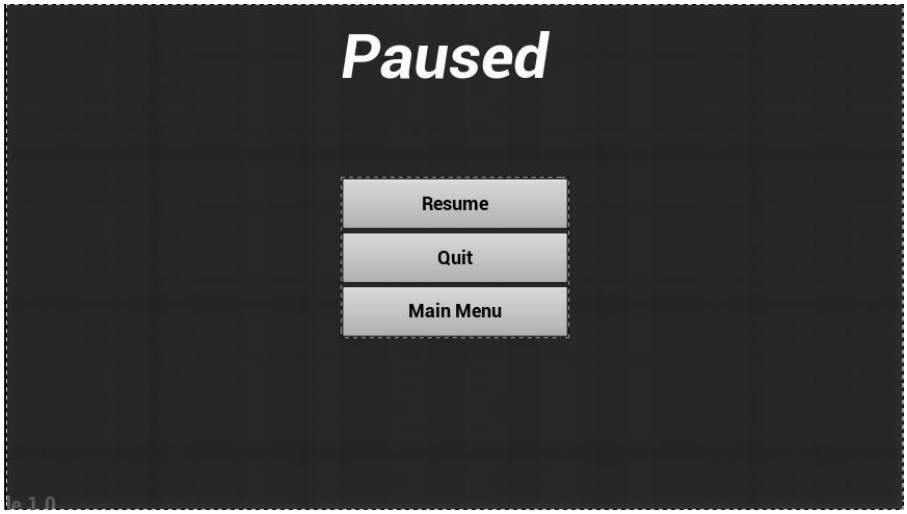


- Open the player character and go to Begin Play.
- Add in a Create Widget node and choose our newly made player UI.
- Promote the return value into a variable which I will call Player Hude.
- Connect this variable to an Add to Viewport node and now our UI will show on screen when we play our game.

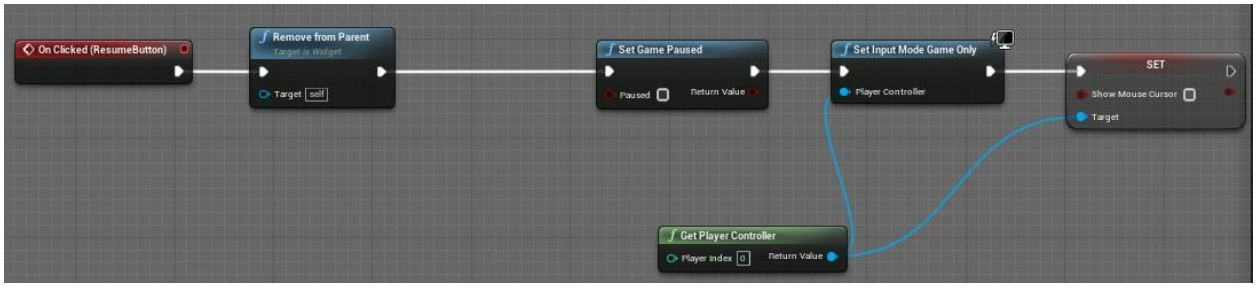


- Now we are going to make a pause menu for our game. Before we start go to the input section of the project settings and make action mapping key for pause which I will set to the P and Escape key.
- Make a Pause menu widget and add a Resume, Quit, and Main Menu button with text. I also added in s screen blur and a big text block that says paused.

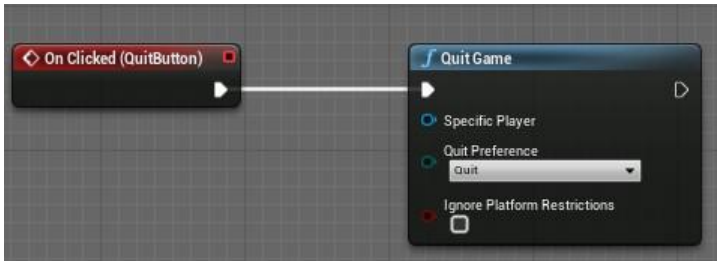
Frogger Styled Game Instructional Guide



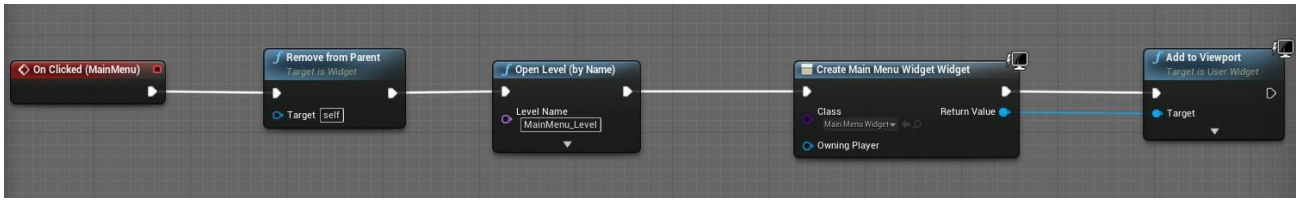
- In the graph add a On Clicked for the resume button. Connect it to a Remove from Parent node and from that connect it to a Set Game Paused node with the paused bool set to false. Add a Get Player Controller and connect to a Set Input Mode Game Only and Set the Show Mouse Cursor to false.



- Add a On Clicked for the quit button and connect it to a Quit Game node.



- Add a On Clicked for the main menu button and from the event connect it to a Remove from Parent node followed an Open Level Node with the level being our main menu map, then a Create Widget node with the widget being our main menu widget, and finally an Add to Viewport node with the target being the return value pin of the Create Widget node.

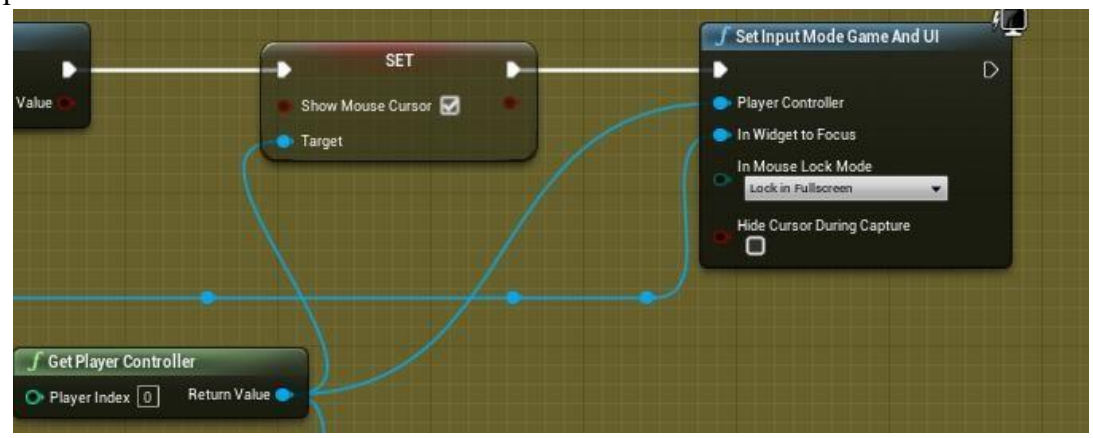


- Go into the player character and add in an InputAction Pause node.

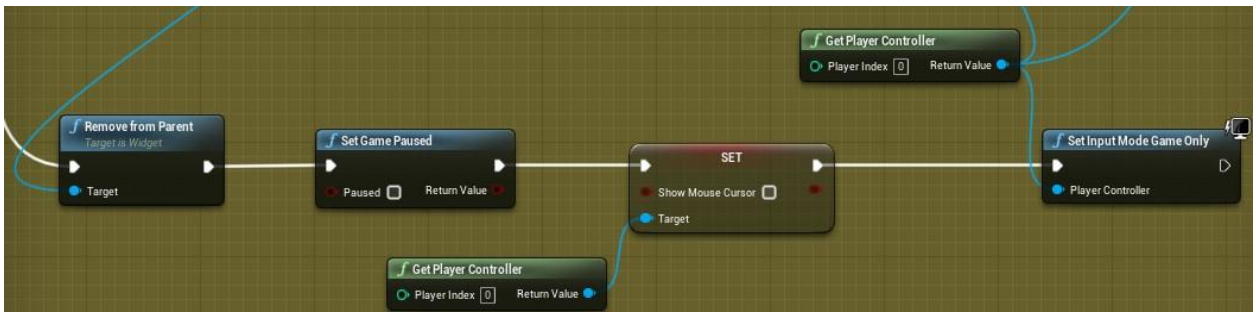
Frogger Styled Game Instructional Guide

- Connect a Flip/Flop node to the pressed pin on the Input Action node.
- Connect a Create Widget node with the main menu as the widget to the A pin of the Flip/Flop node. Add the widget to the viewport with the Add to Viewport node.
- After the Add to Viewport node connect a Set Game Paused node with the Paused bool set to true.

- Next, we are going to get the player controller and use the return value pin to set the Show Mouse Cursor variable to true.
- Drag off the Return Value of the Get Player Controller pin and get the Set Input Mode Game and UI node. Connect the return value of the Create Widget node to the In Widget to Focus pin.

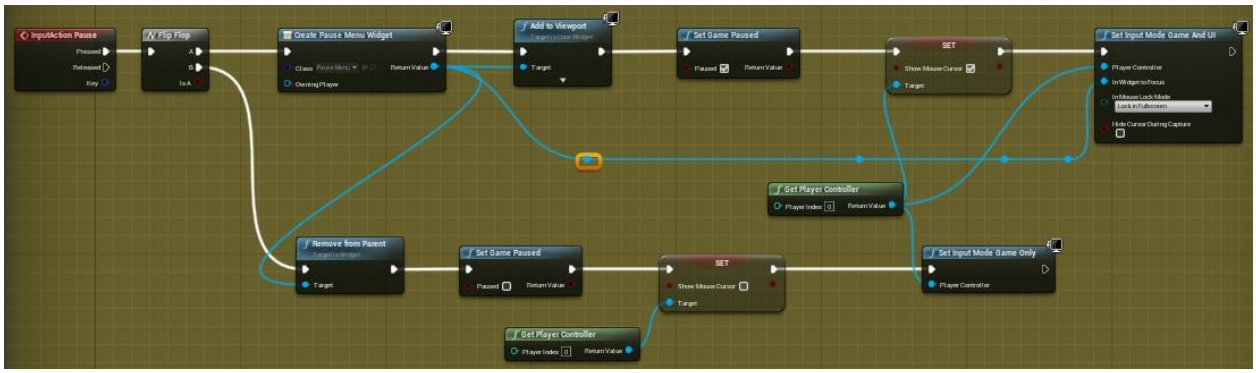


- Connect a Remove from Parent node with the target being the return value of the Create Widget node to the B pin of the Flip/Flop node.
- After the Remove from Parent node connect it to a Set Game Paused node.
- Similar to before, use a Get Player Controller node to get the Show Mouse Cursor variable and set the bool value to false.
- Connect that to a Set Input Mode Game Only node with the player controller pin being connected to the Get Player Controller node.



- With that we now have a working pause, main menu, and working player UI.

Frogger Styled Game Instructional Guide



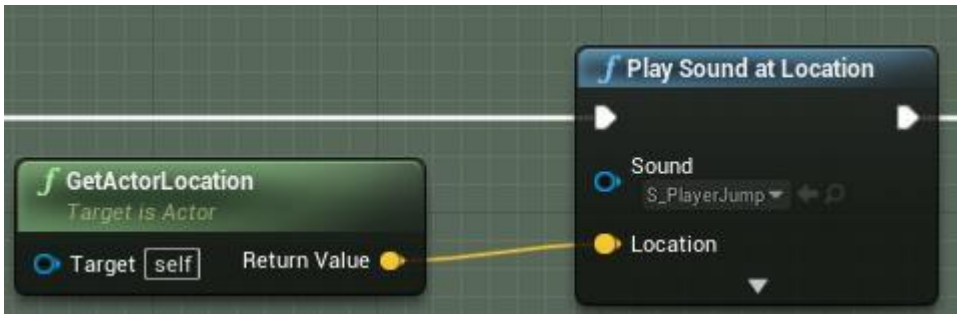
Part 8: Adding Polish

Now that we have our core systems, UI, obstacles, and levels done we can move on to adding polish like sound and visual effects.

Sound Effects

For this section, I will go over how to add sound effects to your game. I will to add a sound effect for our player jump. Unreal Engine will have sounds that you can use but if you want different sounds then you can find some on websites like freesound.org. If you want to download more sounds, make sure that they are WAV files.

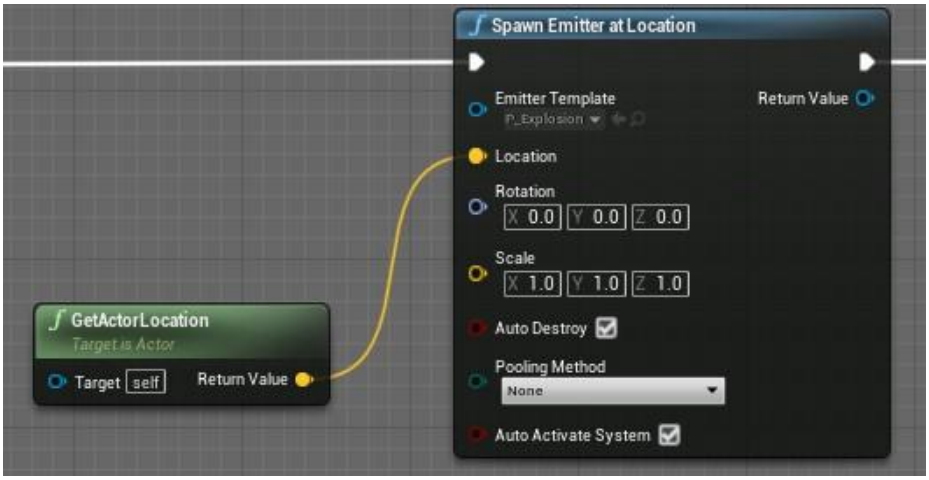
- To add a sound effect simply add in a Play Sound at Location node and choose a sound from the drop menu. Then for the location use the GetActorLocation node.



Frogger Styled Game Instructional Guide

Visual Effects

- To add a visual effect like a particle emitter, use a Spawn Emitter at Location node with the Location being our actor location. I want this emitter to be for the player death so I'm going to use the P_Explosion particle.



Conclusion

If you followed the previous steps, you should have a Frogger like with your own creative spin. You don't have to stop here, if you want you can add in more mechanics and features to create a more complex and original game. If you want to learn how to do more to do more with Unreal Engine then Full Sail, YouTube, and the Unreal Engine site will have plenty of information for you to use. Hopefully this has been a fun and educational experience for all those reading and coding.