GameDev Diaries

Subtitle and Narration System

Nadav Hekselman

Youtube

Define the widget blueprints. Make a widget blueprint names SubtitlesBP. Insert a text component and named the variable as Subtitle. Anchored it to the bottom centre of the screen and centred the alignment and also checked the size to content option so it changes its size according to the text. Gave it the required font, colour and size.

The idea is to control the value of the text box according to the line in the dialog. So checked the isVariable box for the text box so we can use it as a variable. In the graph added a custom event node called Display line and added a reference of the Subtitle variable extended it and added the set text node and hooked its input exec pin to the display line’s output exec pin. Now pulled the pin from the input text and hooked it to the custom event and a new variable pin gets created in the custom event node. Renamed this pin as line to show. The plan is to call the event from other classes and pass the line as parameter and this line is set as the new text.

Now we have the subtitle display functionality. Next, we need to create the subtitle manager which takes care of adding, counting etc of the dialogs. Made a new blueprint class of actor type and named it SubtitlesManager. Opened it and went to the event graph. Extended the event begin play node and added a create widget node and set the class as SubtitlesBP. Extended the return value and promoted it as a variable and named it SubtitlesWidgetReference. Extended the return pin and added an add to viewport node.

To test this much we add the SubtitlesManager anywhere in the level and press play. We see Text Block (which is the default value) written on the screen. But we don’t want the default text to be printed on the screen we want the text to be displayed only when we want it to. So we create a new custom event called ClearSubtitles. Added in a reference of the SubtitlesWidgetReference and extended it and added a display line node. Hooked its input exec pin to the ClearSubtitles node’s output exec pin. We don’t give any value in the input field of Line to show. So whenever this event is called the line to show value is null so no subtitles are displayed. We need to call this at the beginning of the program to clear the subtitles screen so we extend the add to viewport node and add a Clear Subtitles node. Test this now and the screen must look blank.

Next, we need to create a list that contains the dialogs we want the subtitles to display. We create a class of Struct type and name it SubtitlesStruct. Struct (structure) is like the building blocks of a list. It kind of tells the list what information it’s going to hold. Opened SubtitlesStruct and we add the types of variables we want the list to hold. We add 2 variables. The first is of text type and we set the variable name as Line to show. The next is going to be of type – Sound Base and set the variable name as NarrationSound. This is the sound file we want to play while showing the dialog.

For now, we add only the functionality to show the dialogs and we will handle the playing of sound files later. Next, we create a list. Insert a data table and when the prompt to pick row structure comes set the structure as SubtitlesStruct. We named the table as ShipComputerSubtitle list (because in the tutorial the instructor uses this for the subtitles of the AI computer in his game). So we have created a table and each row of the table has the structure of SubtitlesStruct. So whenever we add a row we have 2 parameters – Line to show & NarrationSound. Make a few entries and remember to give the index no: as the name for each row so we can call it using iteration. Just give some text in the Line to show fields and leave the NarrationSound blank for now.

Next, we create a level sequence. A level sequence is a timeline that we use to play animations or events, etc in our level. We create a new animation > level sequence and name it ShipComputerNarrationLS. For each narration sequence we create a new level sequence. For now, let’s just make one with only one audio file. Drag in the audio file into the level sequence. The plan is to trigger the displaying of dialogs at certain points of the audio sequence.

I want the sequence to play whenever the level begins so I opened the level blueprint and extended the event begin play and to easily create a reference to ShipComputerNarration, I click on the instance of it which is in the level, went back to the level blueprint, right-clicked and added a reference to the ShipComputerNarration. This trick is only applicable in level blueprints. Only in it can we click on objects in the scene and bring in references of it in the level BP. Extended the reference and added a play (sequence player) node and hooked the output pin of the begin play node and the input pin of the play node. Now the sequence fires up whenever the level begins.

Now go to the sequence editor, add track > actor to sequence > SubtitleManager. For this first make sure that the SubtitleManager is somewhere in the level. Now in the SubtitleManager add a track > Add event. Now we can just call the event in the timeline whenever we want to display the subtitle.

Right now, there is no way we can call events from the SubtitleManager. We can make a function in it to call events so open the SubtitleManager and make a new function called ShowLine. In the event graph of the SubtitleManager make a custom event node called ShowNextLine. This is the event that’s going to call the next line. We create the function so that it can be used in the level sequence. The function just triggers (or calls) the event as the event can’t directly be called in the level sequence (the instructor isn’t sure about this rule though). Open the ShowLine function extend its output exec pin and add a ShowNextLine node.

To check if it’s working extend the ShowNextLine node and add a print string node. Go to the level sequence at the reqd. point in the timeline add a SubtitleManager event key. Right click on the created event key go to properties and go to event > create quick binding and we can see all the functions of the SubtitleManager that are available to us. Select ShowLine function. So now when we press play the sequence plays and at the event the ShowLine is triggered and that triggers ShowNextLine which calls the print string function. Just copy the event key in the sequence and paste it across the timeline no need to create a new quick binding again.

Next, we create the logic to display the subtitles with the trigger of each event. Go inside the SubtitlesManager and remove the print string node connected to ShowNextLine node. We want the system to know if it’s the first line of the sequence or not. It’s important as the first line is treated differently and if we have multiple sequences, we might need to reset the line to the first line. To do that we need the ShowNextLine to have a new input. Add a new input of Boolean type called First Line. Extend the New Line pin and add a branch node. Make a variable of int type called CurrentRow which is going to keep track of which line we’re at. The default value must be 0. Because we use the Show Line function to call the event and the event now has a New Line input, we need the function to have an input parameter too. So, go to the function blueprint and drag the First Line pin into the function node and a new parameter gets created and name it First Line.

Go back to SubtitleManager and go to the ShowNextLine event add a new input pin of type Data Table and name it NarrationList. Extend the pin and promote it to a new variable called CurrentNarrationList and add it between the event node and the branch node. Do the same for the function too. Drag the event’s narration list input pin and attach it to the function node so that a new input pin gets created on the node.

Extend the true pin of the branch node and add a set CurrentRow node and set the value as 0. Get a reference to the CurrentNarrationList and extend the pin and add a get data table row names node. This node returns all the rows in the data table as an array. Extend the row names pin and add a get node. To this we need to give the index as the first line is the 0th entry type 0 or for better understanding get a reference to the CurrentRow and hook it as the index. Extend the get node and add a get data table row node and to the data table pin hook a reference of CurrentNarrationList. Hook the input exec pin of this node to the output exec pin of the Get Data Table Row Names node.

As each row is a struct containing text and audio, we need to break this struct into the components so extend the out row pin and add a break subtitles struct node. Now that we have the text, we need to display it on screen. Get a reference to the SubtitlesWidget, extend it and add the Display Line fn node and hook the Line to Show pin to the Display Line’s Line to Show pin. Hook the Get Data Table Row’s output exec pin to the Display Line’s input exec pin.

Now that we’ve displayed the first line, we need to tell the computer that we’re no longer on the first line. For this just increment the CurrentRow variable. For this use an increment operator node or just add a set CurrentRow node, extend the input pin and add an int+int node and to one hook a reference to the CurrentRow node and to the other pin set it as 1. Hook the CurrentRow input exec pin to the Display Line node’s output exec pin.

If it’s not the first line just display the line on the row whose value is stored in CurrentRow variable. Get a reference to CurrentNarrationList and extend it and add a get data table row names node and hook its input exec pin with the false pin of the branch node. Extend the out row names pin and add a get node. Add a get CurrentRow node and hook it to the index pin of the get node. Extend the get node and add a get data table row node and add a get CurrentNarrationList and hook it to the Data Table pin and hook the get data table row node’s input exec pin with the get data table row names node’s output exec pin. Extend the out row node and add a break subtitles struct node and add a get SubtitlesWidget node and extend it and add a Display Line fn node and extend the Line to Show pin of the break node with the Display Line’s Line to Show pin and hook the Display Line node’s input exec pin with the get data table row node’s output exec pin. Finally extend the Display line fn node’s output exec pin and add a set CurrentRow node and extend it’s input pin and add an int+int node. To one of it add a reference to the CurrentRow and to the other pin set the value as 1.

Now that we have the whole functionality set up we need a way to trigger these in the event track. Go to the level sequence and right click the event key > properties >create binding and choose Show Line. The ShowLine 1 BP appears (as this is the 1st event key, for this key the function instance is called ShowLine1) and as it’s the first line check the First Line box and also set the NarrationList as ShipComputerSubtitleList. This much will show the first subtitles. To show the rest make another event key right-click > properties > event binding > ShowLine. The ShowLine 2 BP appears and as it’s the 2nd line set First Line box to false and set the list. For the rest of the lines you don’t have to make separate keys as the functionality automatically selects the row indexes after the first row the same event key works for the rest as well so just copy and paste them through the timeline.

We can further modify this such that the function takes in numbers as input and display the appropriate row number this removes the necessity of multiple lists although might take up a bit more memory but that’s insignificant I think.