DIALOG TUTORIAL FOR NOOBS V3

By: Iceman77

Being a novice scripter myself, coupled with the stigma that dialogs are in general, just plain evil, I found it pretty frustrating to start making dialogs. There are no viable tutorials. There are no walkthroughs. Just the wiki. While technically, there is a small hello world tutorial, this leaves a lot of room for wonder. Especially for noobs. Most people who contribute to the wiki are seemingly in some l33t scripter's contest, instead of contributing in a practical way. This guide is intended for anyone wanting to learn how to make a basic yes / no dialog, with a picture and a few words. This guide is not meant for advanced users. If you're a dialog guru, then stop reading. I'm sure you would do everything differently. **Just notify me when you make a dialog tutorial for noobs and I'll take this one down.** In any case, this guide will however, get a noob started out making his or her own, working UI dialogs.

If you don't know how to run addons, install & preview a mission in the editor, manage script files or convert jpg/png pics to .paa, then I seriously suggest closing this dialog tutorial and learn how to do those things. As it's possible that this may be above your head, if you can't download & install addons or don't know how to manage scripts, mission files & pictures.

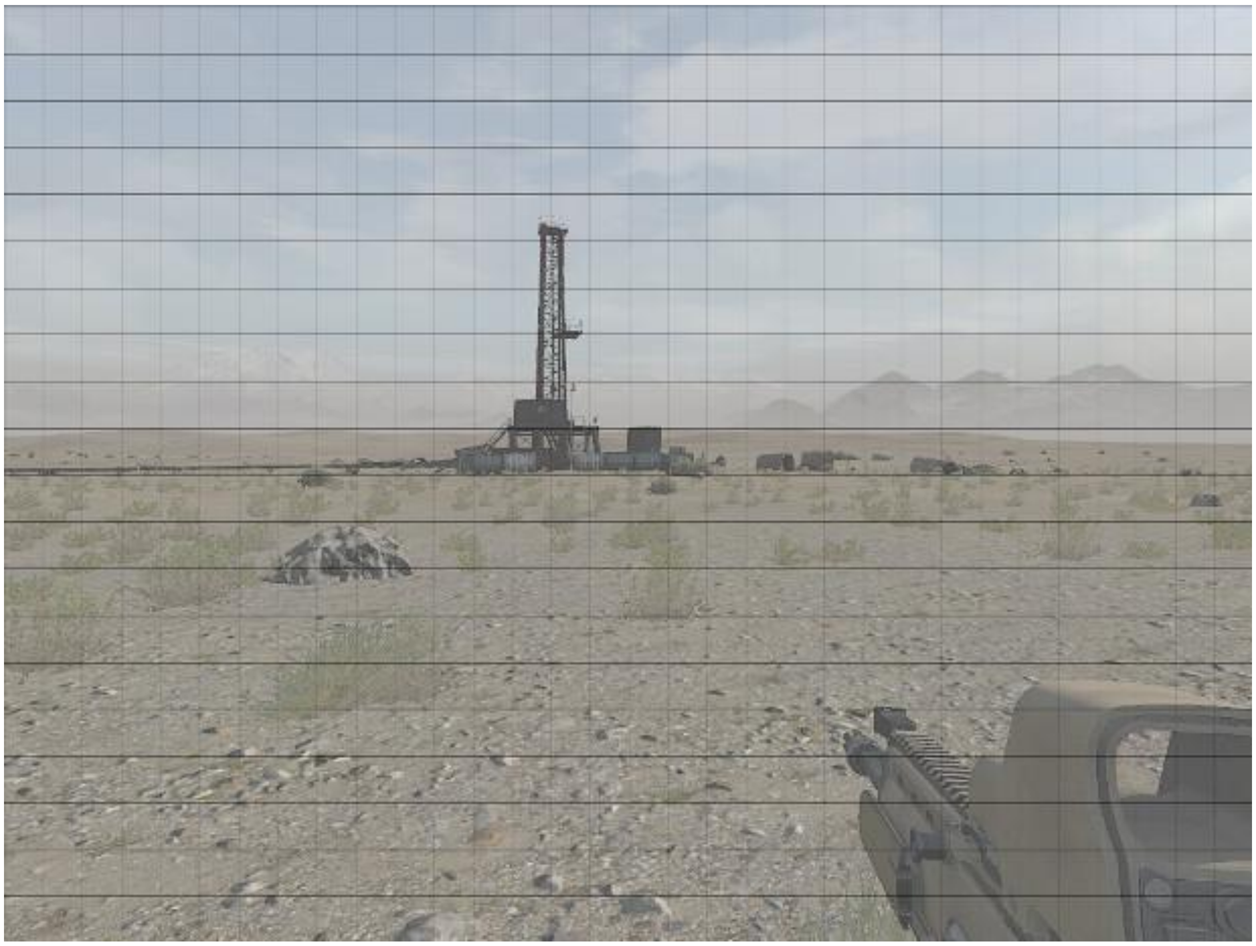
Okay, so before you begin creating a dialog, create a new mission folder by starting up the game, going into the editor and saving a new mission on Desert. Always use desert when testing scripts, dialogs w/e. It loads up the quickest. Once the mission is saved, create an blank dialogs.hpp, defines.hpp & an description.ext in the mission folder. Exit out of the game. We'll need the mission folder you created way later on, towards the end of the tutorial.

Go download the GUI editor addon & the GUI editor preview mission @ **Download GUI editor** **&** Preview Mission

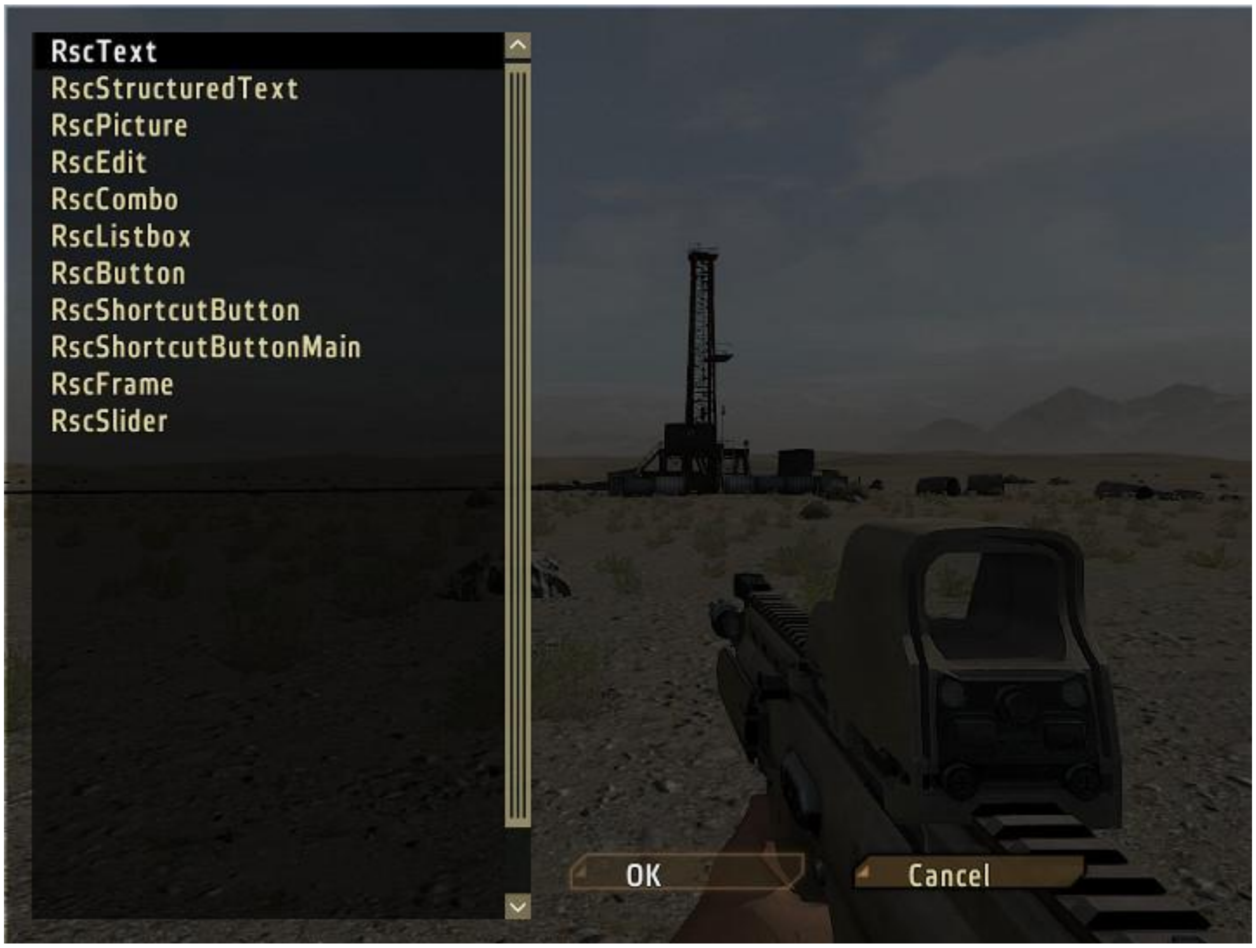
IMPORTANT!!! BELOW IS MY PERSONAL EXAMPLE MISSION FILE. DON'T COPY & PASTE CODE FROM MY EXAMPLE MISSION!!! USE THE RESOURCES PROVIDED IN THE SPOILERS, WHICH RESIDE IN THE MAIN THREAD "DIALOG TUTORIAL FOR NOOBS" IN THE EDITING AND SCRIPTING SCETION.

http://arma2.co/download?file=82265-ICE_DIALOG.Desert_E.rar

Note: the GUI addon isn't required to use the dialog we're going to make. It's just to have access to the GUI tool. You run the addon just like any other addon. Once you have the addon & the preview mission installed, start up OA/CO. Load the preview mission in the editor and preview it. Use your radio (0-0-1) to call the GUI Editor. You should have something like the image below. **Note:** You can hit H to bring up the help menu.



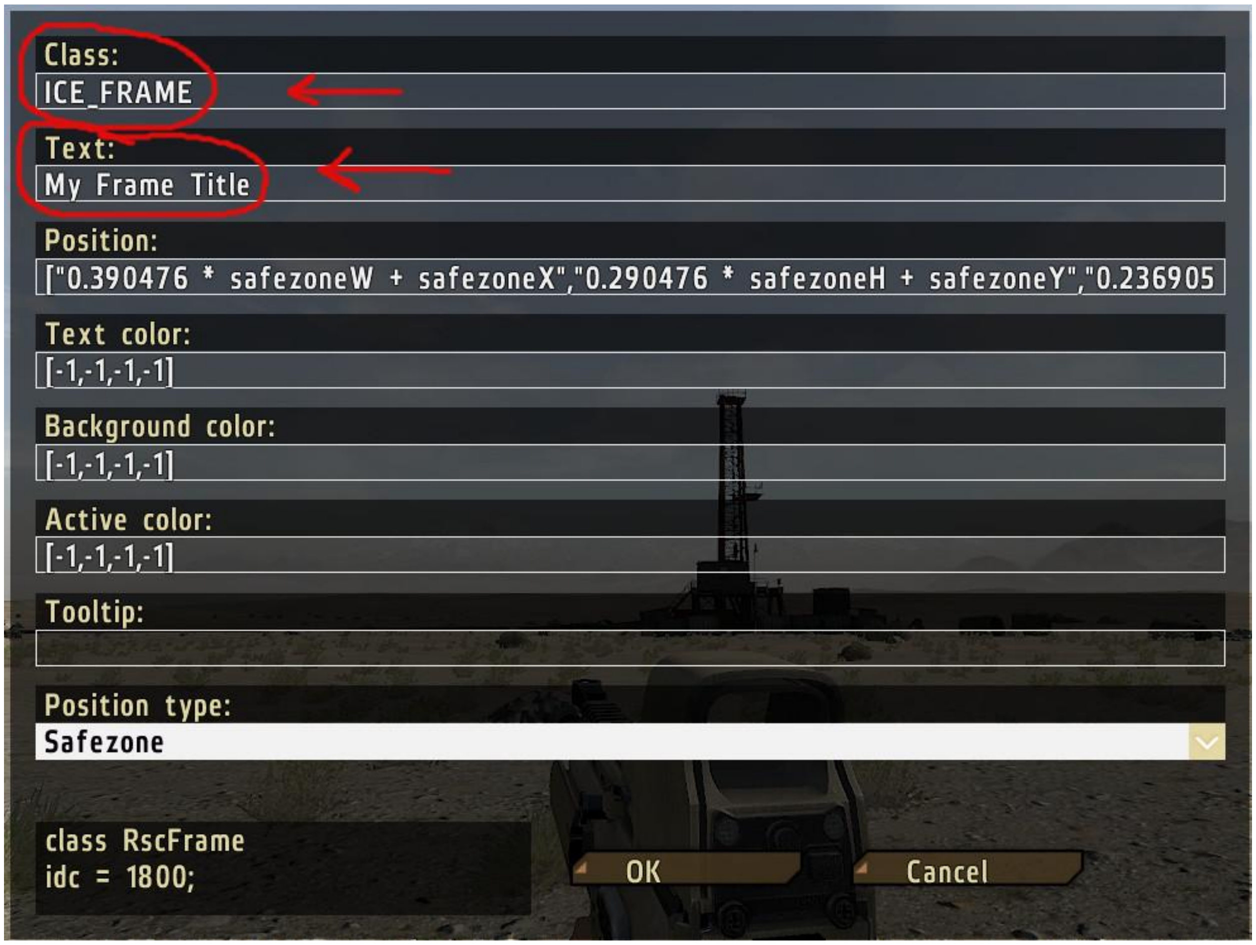
Okay, this is the fun part. Right click on the grid and this UI will come up:



We'll start with a frame first just so we can determine roughly how big our dialog is going to be. Click on RscFrame & hit OK.

You'll End up with a small rectangle on the grid. You can hold ALT & left click on the frame to scale it how you want it. You can simply just **<u>move</u>** the frame by left clicking on the frame and positioning it where you want it (pref. centered on the grid).

You can right click on the frame and edit some properties. We will not touch any properties here except the Text & Class properties. Each piece of the dialog you create, you'll assign a unique, but consistent class name. Throughout this part of the tutorial, I'll start all of my unique class names with ICE. In example; ICE_FRAME. You can name it whatever you like, but to keep things consistent, you should come up with a neat, uniform naming convention. So things are easily read, understood & edited at a later date. Also you can give the Frame some text. Whatever you like. Once you've assigned a unique class name and some text hit OK. Take a look at the image below for further reference to class name & Text.

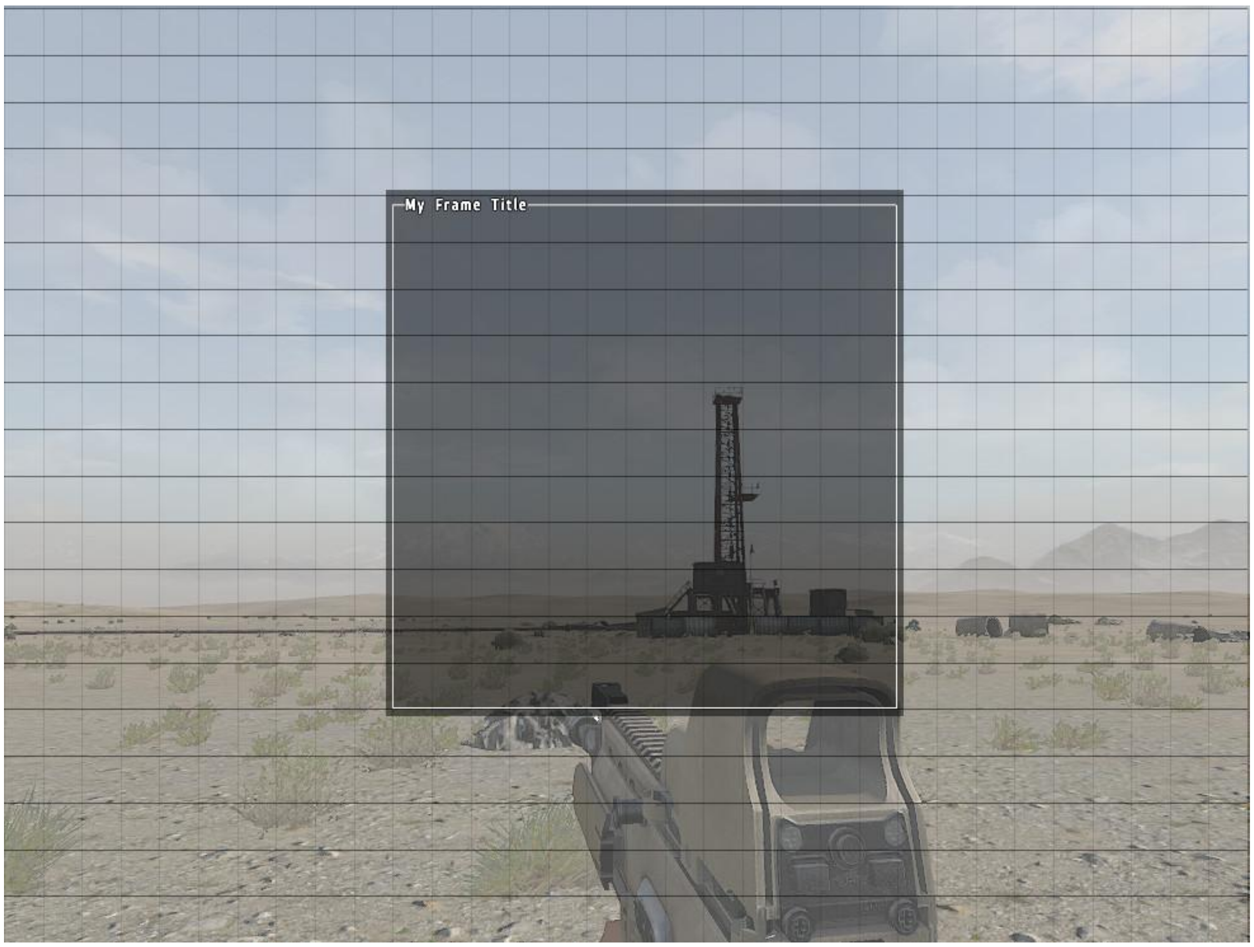

Class:
ICE_FRAME
Text:
My Frame Title
Position:
["0.390476 * safezoneW + safezoneX","0.290476 * safezoneH + safezoneY","0.236905
Text color:
[-1,-1,-1,-1]
Background color:
[-1,-1,-1,-1]
Active color:
[-1,-1,-1,-1]
Tooltip:

Position type:
Safezone

```
class RscFrame
idc = 1800;
```

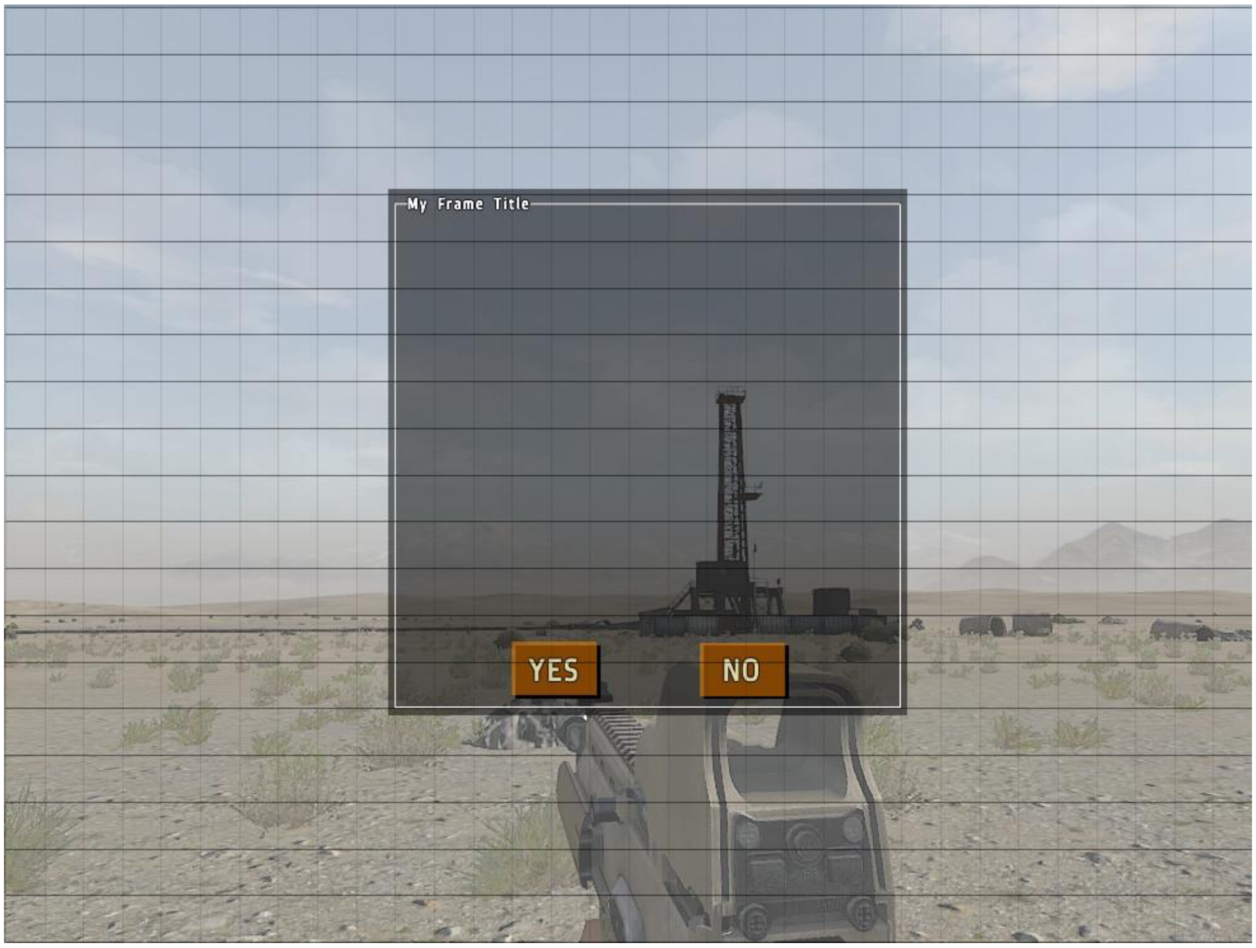
Okay now, if you've created and edited your frame it should look something like the image below.

Like so:

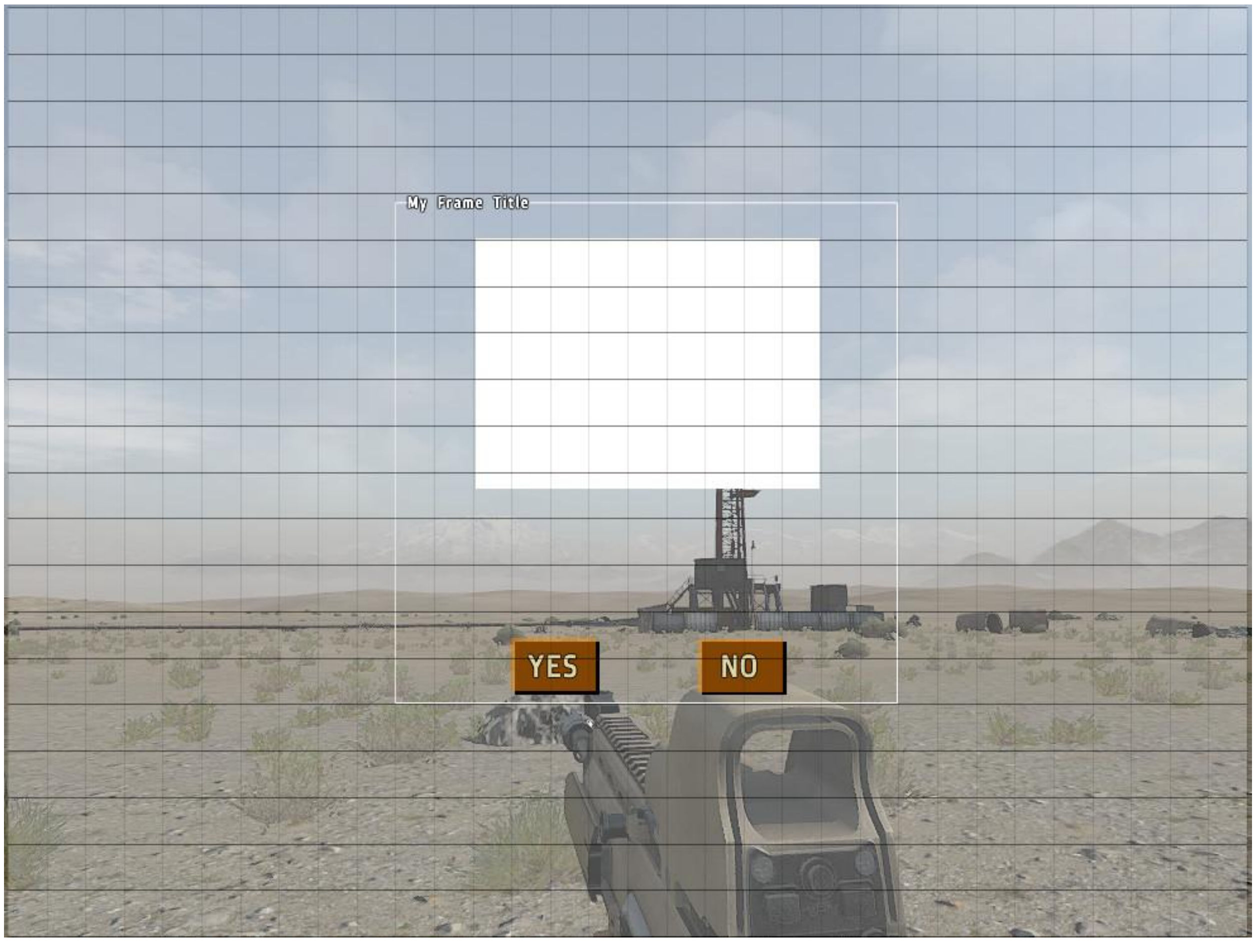


Once you have a size you want, it's positioned where you want (pref. close to the center) & you've set the text & class properties, right click on an empty part of the grid again to bring up the UI.

Now we'll add a couple buttons. Select rscButton & hit OK. While mousing over the newly created button, hit CTRL+C to copy the button. Hit CTRL+V to paste a new button. Position them how you want within the frame, and once again, If you're not satisfied with the size of the button, you can hold ALT + Left Click to scale the button to the size you want. Right click each button to edit the text & class properties for each. For one button simply put YES in the text field. For the other put NO in the text field. Also, give each button a unique class name.IE; ICE_ButtonYes & ICE_ButtonNo. Remember proper class naming convention (above). You now should have something like the image below.

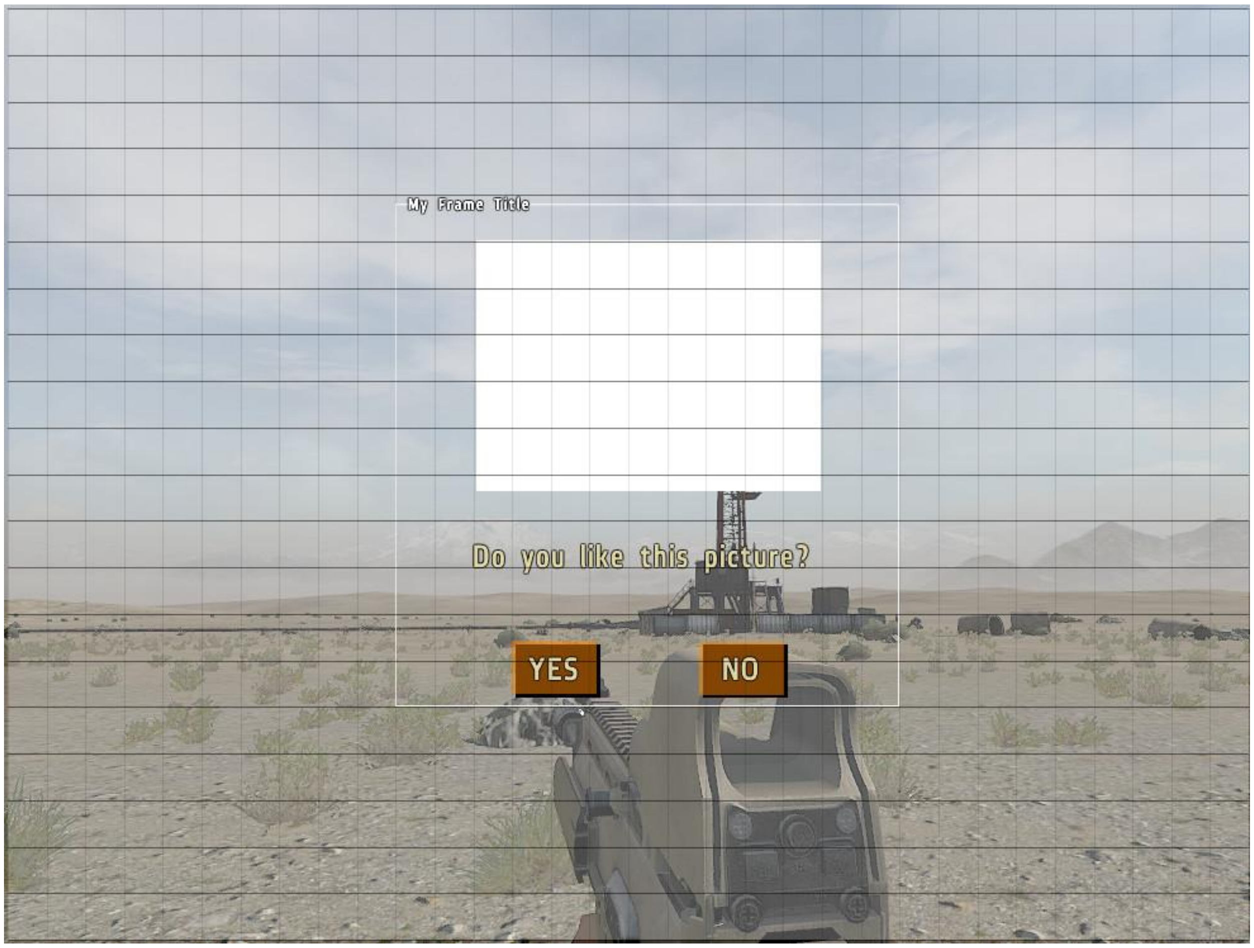


Okay so now we'll add a picture space. Right click on an empty part of the grid to bring up the UI. Click on rscPicture & hit okay. Scale it accordingly (ALT+LEFT CLICK DRAG) & place it where you want it to sit in the UI. Assign a unique class name to the picture.IE; ICE_PICTURE. You don't have to include any text, seeing how it's a picture. You should now have something like the image below

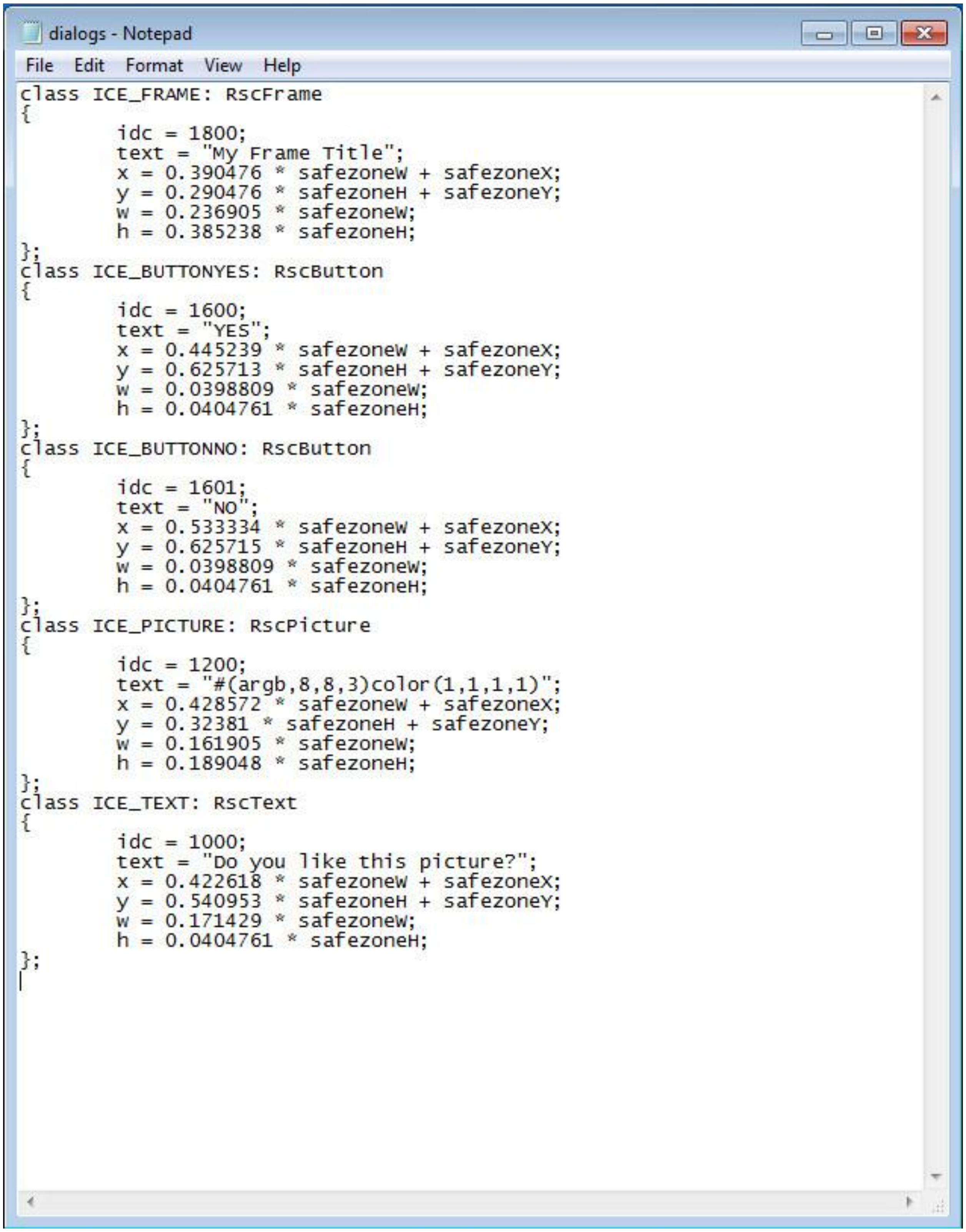


We’ll now assign some text on the actual dialog. Right click on an empty space on the grid to bring up the UI. Select rscText & hit OK. Scale the text box to a proper size (ALT+LEFT CLICK DRAG). Give it a unique class name IE; ICE_TEXT. Also, guess what else??!! Give it some text!!

Place the text where you want within the Frame. You should now have:

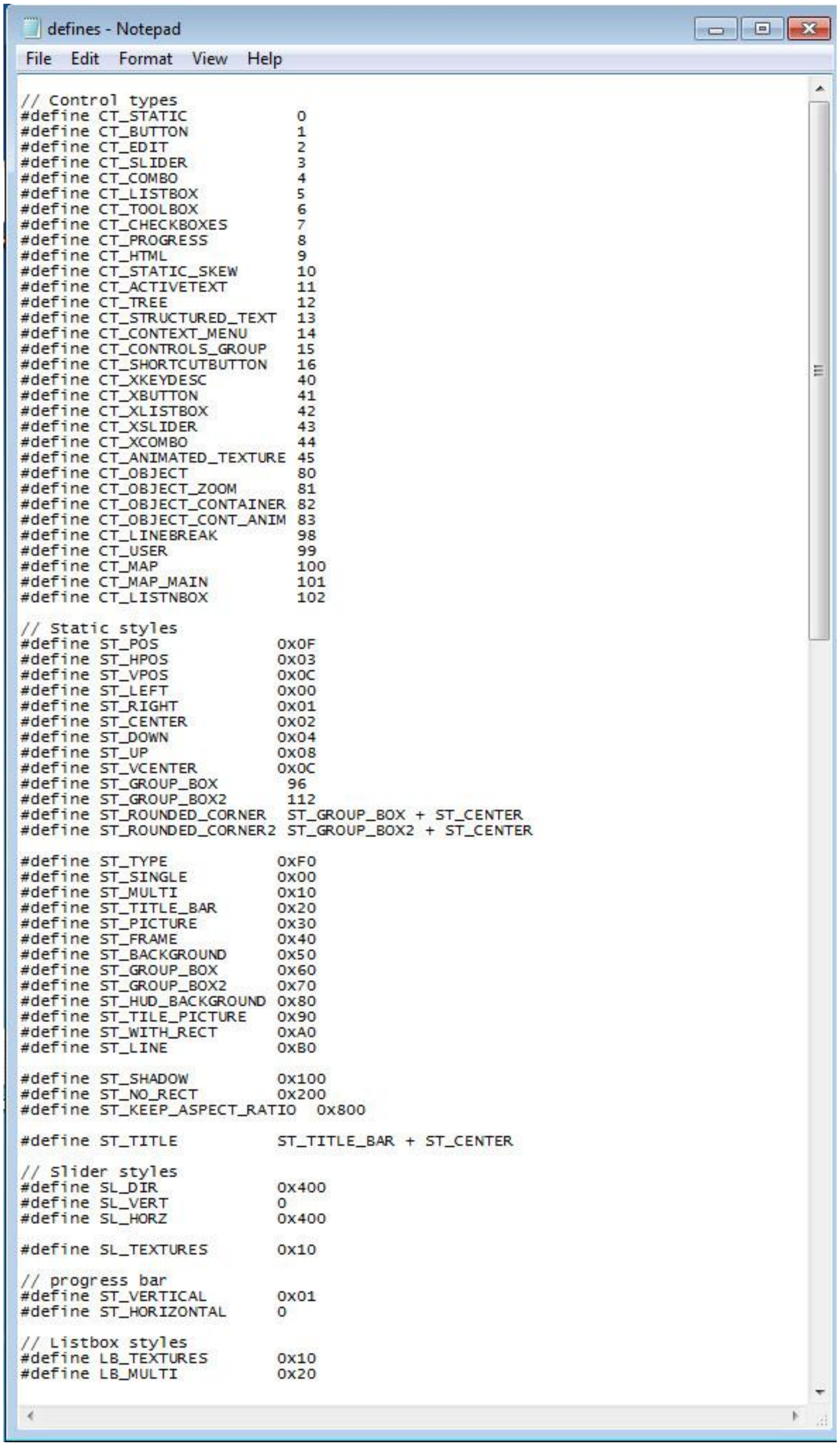


Okay, now we’re going to hit CTRL + SHIFT + S to “copy” our cool new dialog, so we can paste what you’ve created into a file. Remember that mission folder with the 3 blank files you created at the start of this tutorial? You did do that right? Go into that mission folder, open the blank dialogs.hpp and hit CTRL + V to paste in your created dialog. The file should look something like the image below.

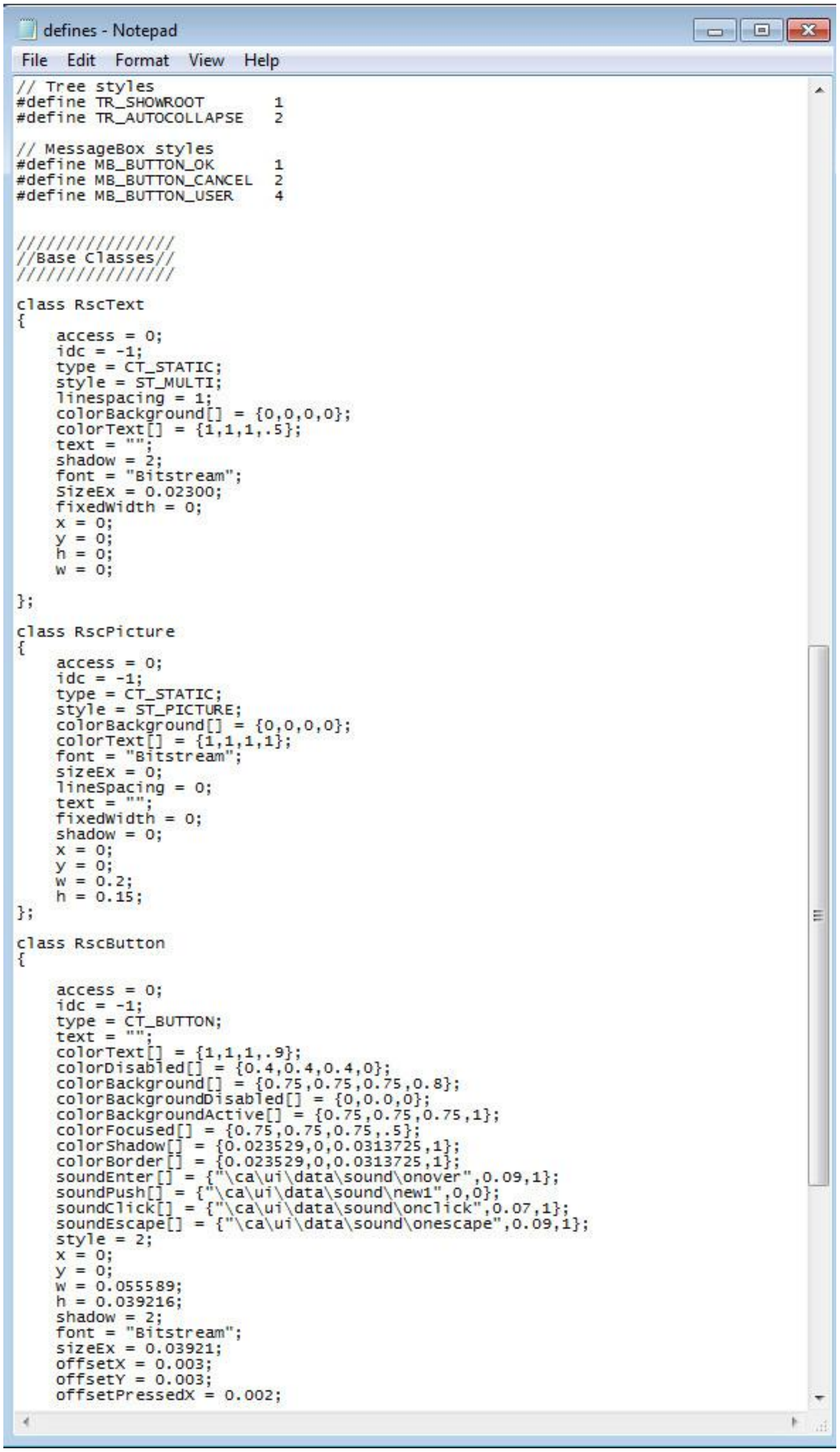


```
class ICE_FRAME: RscFrame
{
        idc = 1800;
        text = "My Frame Title";
        x = 0.390476 * safezoneW + safezoneX;
        y = 0.290476 * safezoneH + safezoneY;
        w = 0.236905 * safezoneW;
        h = 0.385238 * safezoneH;
};
class ICE_BUTTONYES: RscButton
{
        idc = 1600;
        text = "YES";
        x = 0.445239 * safezoneW + safezoneX;
        y = 0.625713 * safezoneH + safezoneY;
        w = 0.0398809 * safezoneW;
        h = 0.0404761 * safezoneH;
};
class ICE_BUTTONNO: RscButton
{
        idc = 1601;
        text = "NO";
        x = 0.533334 * safezoneW + safezoneX;
        y = 0.625715 * safezoneH + safezoneY;
        w = 0.0398809 * safezoneW;
        h = 0.0404761 * safezoneH;
};
class ICE_PICTURE: RscPicture
{
        idc = 1200;
        text = "#(argb,8,8,3)color(1,1,1,1)";
        x = 0.428572 * safezoneW + safezoneX;
        y = 0.32381 * safezoneH + safezoneY;
        w = 0.161905 * safezoneW;
        h = 0.189048 * safezoneH;
};
class ICE_TEXT: RscText
{
        idc = 1000;
        text = "Do you like this picture?";
        x = 0.422618 * safezoneW + safezoneX;
        y = 0.540953 * safezoneH + safezoneY;
        w = 0.171429 * safezoneW;
        h = 0.0404761 * safezoneH;
};
```

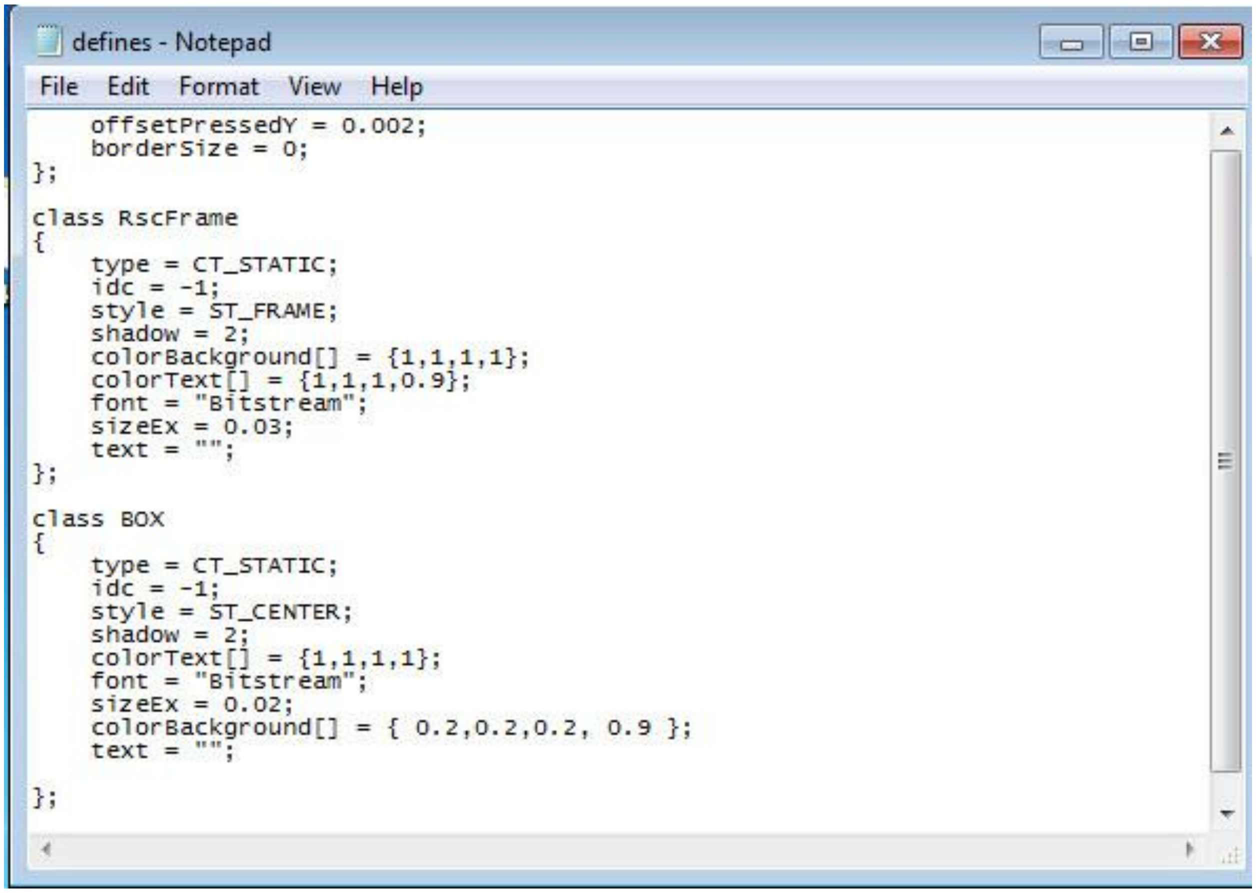
Okay now that we have our actual dialog pasted into the dialogs.hpp, save and close that file. Now we have to create some defines & base classes. Basically we have to let the game know what types of UI elements to expect / are gonna be created. Now usually, there is a typical set of defines dialog makers copy & paste from one dialog to another. A lot of the time, a good portion of the defines are un needed, but it's still good to have them there for future edits. IE; all of the sudden you decide to add a scroll menu to the dialog. The define is already there =). Open up the blank defines.hpp & Copy n' paste the defines & base classes from the resources provided into your defines.hpp. These defines & base classes, along with all files can also be found @ the official BI forums "Dialog Tutorial For Noobs" Thread, in the editing and scripting section. Your defines.hpp should look like the images below (defines.hpp split into multiple images)



```
// Control types
#define CT_STATIC            0
#define CT_BUTTON            1
#define CT_EDIT              2
#define CT_SLIDER            3
#define CT_COMBO             4
#define CT_LISTBOX           5
#define CT_TOOLBOX           6
#define CT_CHECKBOXES        7
#define CT_PROGRESS          8
#define CT_HTML              9
#define CT_STATIC_SKEW       10
#define CT_ACTIVETEXT        11
#define CT_TREE              12
#define CT_STRUCTURED_TEXT   13
#define CT_CONTEXT_MENU      14
#define CT_CONTROLS_GROUP    15
#define CT_SHORTCUTBUTTON    16
#define CT_XKEYDESC          40
#define CT_XBUTTON           41
#define CT_XLISTBOX          42
#define CT_XSLIDER           43
#define CT_XCOMBO            44
#define CT_ANIMATED_TEXTURE  45
#define CT_OBJECT            80
#define CT_OBJECT_ZOOM       81
#define CT_OBJECT_CONTAINER  82
#define CT_OBJECT_CONT_ANIM  83
#define CT_LINEBREAK         98
#define CT_USER              99
#define CT_MAP               100
#define CT_MAP_MAIN          101
#define CT_LISTNBOX          102

// Static styles
#define ST_POS             0x0F
#define ST_HPOS            0x03
#define ST_VPOS            0x0C
#define ST_LEFT            0x00
#define ST_RIGHT           0x01
#define ST_CENTER          0x02
#define ST_DOWN            0x04
#define ST_UP              0x08
#define ST_VCENTER         0x0C
#define ST_GROUP_BOX        96
#define ST_GROUP_BOX2       112
#define ST_ROUNDED_CORNER   ST_GROUP_BOX + ST_CENTER
#define ST_ROUNDED_CORNER2 ST_GROUP_BOX2 + ST_CENTER

#define ST_TYPE            0xF0
#define ST_SINGLE          0x00
#define ST_MULTI           0x10
#define ST_TITLE_BAR       0x20
#define ST_PICTURE         0x30
#define ST_FRAME           0x40
#define ST_BACKGROUND      0x50
#define ST_GROUP_BOX       0x60
#define ST_GROUP_BOX2      0x70
#define ST_HUD_BACKGROUND  0x80
#define ST_TILE_PICTURE    0x90
#define ST_WITH_RECT       0xA0
#define ST_LINE            0xB0

#define ST_SHADOW          0x100
#define ST_NO_RECT         0x200
#define ST_KEEP_ASPECT_RATIO   0x800

#define ST_TITLE           ST_TITLE_BAR + ST_CENTER

// Slider styles
#define SL_DIR             0x400
#define SL_VERT            0
#define SL_HORZ            0x400

#define SL_TEXTURES        0x10

// progress bar
#define ST_VERTICAL        0x01
#define ST_HORIZONTAL      0

// Listbox styles
#define LB_TEXTURES        0x10
#define LB_MULTI           0x20
```

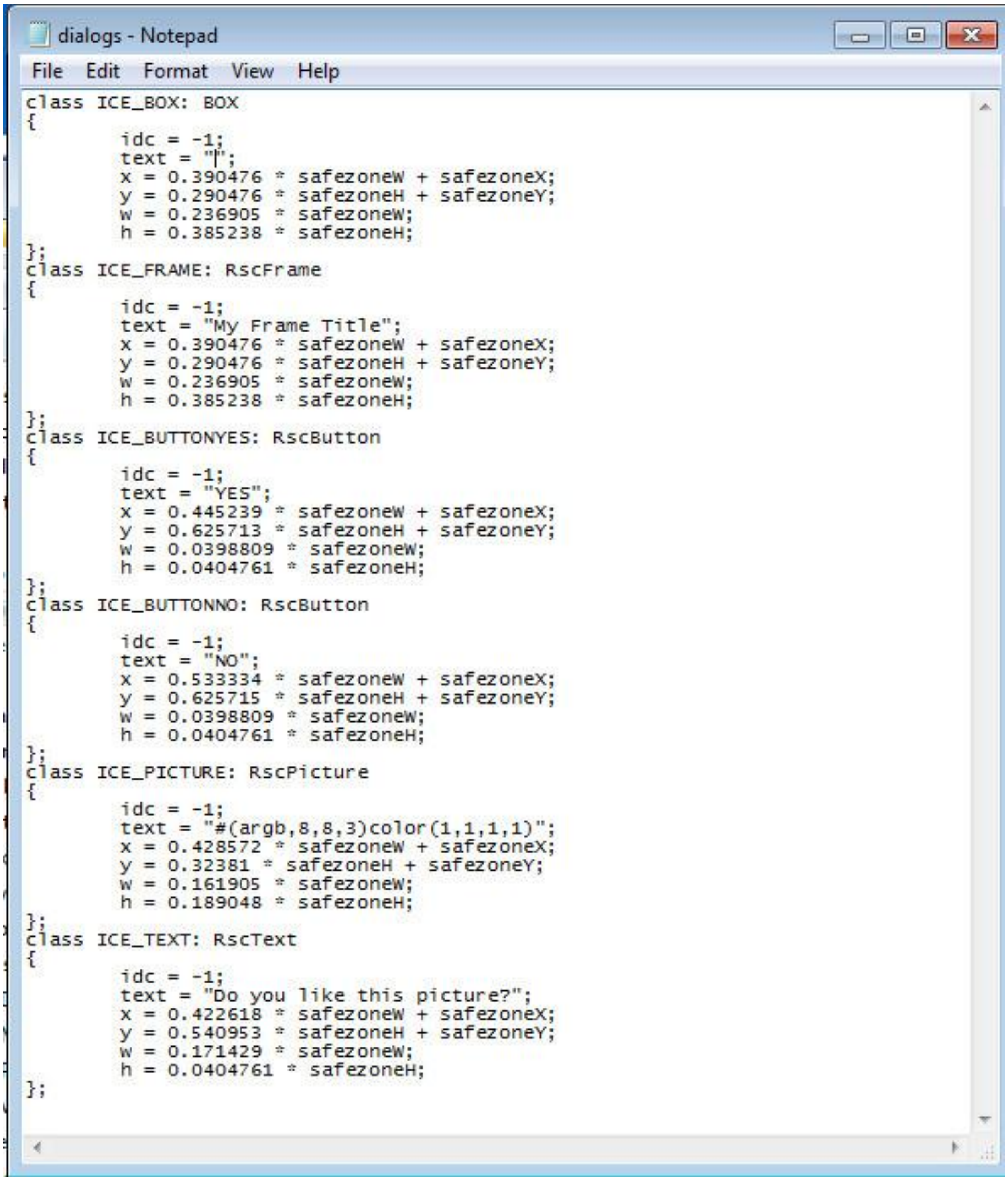


```
// Tree styles
#define TR_SHOWROOT        1
#define TR_AUTOCOLLAPSE    2

// MessageBox styles
#define MB_BUTTON_OK       1
#define MB_BUTTON_CANCEL   2
#define MB_BUTTON_USER     4


////////////////
//Base Classes//
////////////////

class RscText
{
    access = 0;
    idc = -1;
    type = CT_STATIC;
    style = ST_MULTI;
    linespacing = 1;
    colorBackground[] = {0,0,0,0};
    colorText[] = {1,1,1,.5};
    text = "";
    shadow = 2;
    font = "Bitstream";
    SizeEx = 0.02300;
    fixedWidth = 0;
    x = 0;
    y = 0;
    h = 0;
    w = 0;

};

class RscPicture
{
    access = 0;
    idc = -1;
    type = CT_STATIC;
    style = ST_PICTURE;
    colorBackground[] = {0,0,0,0};
    colorText[] = {1,1,1,1};
    font = "Bitstream";
    sizeEx = 0;
    lineSpacing = 0;
    text = "";
    fixedWidth = 0;
    shadow = 0;
    x = 0;
    y = 0;
    w = 0.2;
    h = 0.15;
};

class RscButton
{

    access = 0;
    idc = -1;
    type = CT_BUTTON;
    text = "";
    colorText[] = {1,1,1,.9};
    colorDisabled[] = {0.4,0.4,0.4,0};
    colorBackground[] = {0.75,0.75,0.75,0.8};
    colorBackgroundDisabled[] = {0,0.0,0};
    colorBackgroundActive[] = {0.75,0.75,0.75,1};
    colorFocused[] = {0.75,0.75,0.75,.5};
    colorShadow[] = {0.023529,0,0.0313725,1};
    colorBorder[] = {0.023529,0,0.0313725,1};
    soundEnter[] = {"\ca\ui\data\sound\onover",0.09,1};
    soundPush[] = {"\ca\ui\data\sound\new1",0,0};
    soundClick[] = {"\ca\ui\data\sound\onclick",0.07,1};
    soundEscape[] = {"\ca\ui\data\sound\onescape",0.09,1};
    style = 2;
    x = 0;
    y = 0;
    w = 0.055589;
    h = 0.039216;
    shadow = 2;
    font = "Bitstream";
    sizeEx = 0.03921;
    offsetX = 0.003;
    offsetY = 0.003;
    offsetPressedX = 0.002;
```



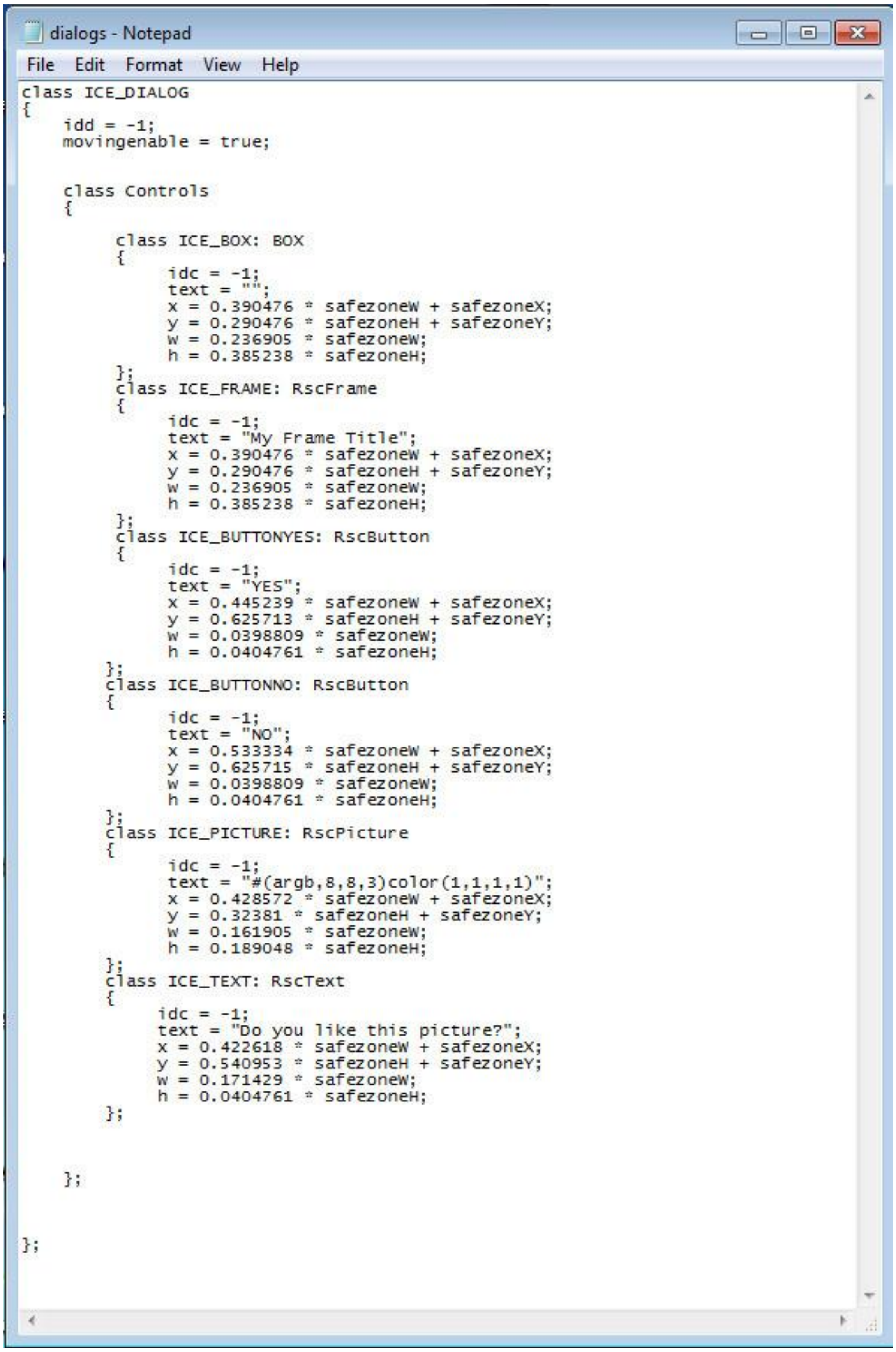
```
    offsetPressedY = 0.002;
    borderSize = 0;
};

class RscFrame
{
    type = CT_STATIC;
    idc = -1;
    style = ST_FRAME;
    shadow = 2;
    colorBackground[] = {1,1,1,1};
    colorText[] = {1,1,1,0.9};
    font = "Bitstream";
    sizeEx = 0.03;
    text = "";
};

class BOX
{
    type = CT_STATIC;
    idc = -1;
    style = ST_CENTER;
    shadow = 2;
    colorText[] = {1,1,1,1};
    font = "Bitstream";
    sizeEx = 0.02;
    colorBackground[] = { 0.2,0.2,0.2, 0.9 };
    text = "";

};
```

You recognize these classes eh? All but one? You don't recognize this BOX class do you? The BOX class serves as our background. We will need to go into our dialogs.hpp and make this class manually also. Okay, lets head into our dialogs.hpp and do that. Once inside the dialogs.hpp, copy the FRAME class and paste it above the FRAME class. Rename the class with a unique name. IE; I named mine ICE_BOX (pun intended). Delete the text from the ICE_BOX class so it's an empty string. This is important as this is our background. Also, we'll set the IDC values to -1 for our entire dialog. The IDC value is used to reference the control class in an external script for example. But since we wont be creating a complex dialog, idcs aren't needed. -1 basically means just that. It serves as a place holder, that means nothing. Your dialogs.hpp should now look like the image below.



```
class ICE_BOX: BOX
{
        idc = -1;
        text = "";
        x = 0.390476 * safezoneW + safezoneX;
        y = 0.290476 * safezoneH + safezoneY;
        w = 0.236905 * safezoneW;
        h = 0.385238 * safezoneH;
};
class ICE_FRAME: RscFrame
{
        idc = -1;
        text = "My Frame Title";
        x = 0.390476 * safezoneW + safezoneX;
        y = 0.290476 * safezoneH + safezoneY;
        w = 0.236905 * safezoneW;
        h = 0.385238 * safezoneH;
};
class ICE_BUTTONYES: RscButton
{
        idc = -1;
        text = "YES";
        x = 0.445239 * safezoneW + safezoneX;
        y = 0.625713 * safezoneH + safezoneY;
        w = 0.0398809 * safezoneW;
        h = 0.0404761 * safezoneH;
};
class ICE_BUTTONNO: RscButton
{
        idc = -1;
        text = "NO";
        x = 0.533334 * safezoneW + safezoneX;
        y = 0.625715 * safezoneH + safezoneY;
        w = 0.0398809 * safezoneW;
        h = 0.0404761 * safezoneH;
};
class ICE_PICTURE: RscPicture
{
        idc = -1;
        text = "#(argb,8,8,3)color(1,1,1,1)";
        x = 0.428572 * safezoneW + safezoneX;
        y = 0.32381 * safezoneH + safezoneY;
        w = 0.161905 * safezoneW;
        h = 0.189048 * safezoneH;
};
class ICE_TEXT: RscText
{
        idc = -1;
        text = "Do you like this picture?";
        x = 0.422618 * safezoneW + safezoneX;
        y = 0.540953 * safezoneH + safezoneY;
        w = 0.171429 * safezoneW;
        h = 0.0404761 * safezoneH;
};
```

While the dialogs.hpp is still open, we need to basically wrap all of this shit up, into a **main class** and **class controls** so we can call/create the dialog (on the screen). You'll name your dialog what you want. In the example I named mine ICE_DIALOG. **See below how to do this.** Once done, save and close the dialogs.hpp.



```
class ICE_DIALOG
{
    idd = -1;
    movingenable = true;

    class Controls
    {

         class ICE_BOX: BOX
         {
              idc = -1;
              text = "";
              x = 0.390476 * safezoneW + safezoneX;
              y = 0.290476 * safezoneH + safezoneY;
              w = 0.236905 * safezoneW;
              h = 0.385238 * safezoneH;
         };
         class ICE_FRAME: RscFrame
         {
              idc = -1;
              text = "My Frame Title";
              x = 0.390476 * safezoneW + safezoneX;
              y = 0.290476 * safezoneH + safezoneY;
              w = 0.236905 * safezoneW;
              h = 0.385238 * safezoneH;
         };
         class ICE_BUTTONYES: RscButton
         {
              idc = -1;
              text = "YES";
              x = 0.445239 * safezoneW + safezoneX;
              y = 0.625713 * safezoneH + safezoneY;
              w = 0.0398809 * safezoneW;
              h = 0.0404761 * safezoneH;
        };
        class ICE_BUTTONNO: RscButton
        {
              idc = -1;
              text = "NO";
              x = 0.533334 * safezoneW + safezoneX;
              y = 0.625715 * safezoneH + safezoneY;
              w = 0.0398809 * safezoneW;
              h = 0.0404761 * safezoneH;
        };
        class ICE_PICTURE: RscPicture
        {
              idc = -1;
              text = "#(argb,8,8,3)color(1,1,1,1)";
              x = 0.428572 * safezoneW + safezoneX;
              y = 0.32381 * safezoneH + safezoneY;
              w = 0.161905 * safezoneW;
              h = 0.189048 * safezoneH;
        };
        class ICE_TEXT: RscText
        {
             idc = -1;
             text = "Do you like this picture?";
             x = 0.422618 * safezoneW + safezoneX;
             y = 0.540953 * safezoneH + safezoneY;
             w = 0.171429 * safezoneW;
             h = 0.0404761 * safezoneH;
        };


    };


};
```

Okay, now we need to go into our description.ext and include the dialogs.hpp & the defines.hpp. This is very simple. Just note the order they are defined, as this is important. Type or Copy and paste what's below into your description.ext.

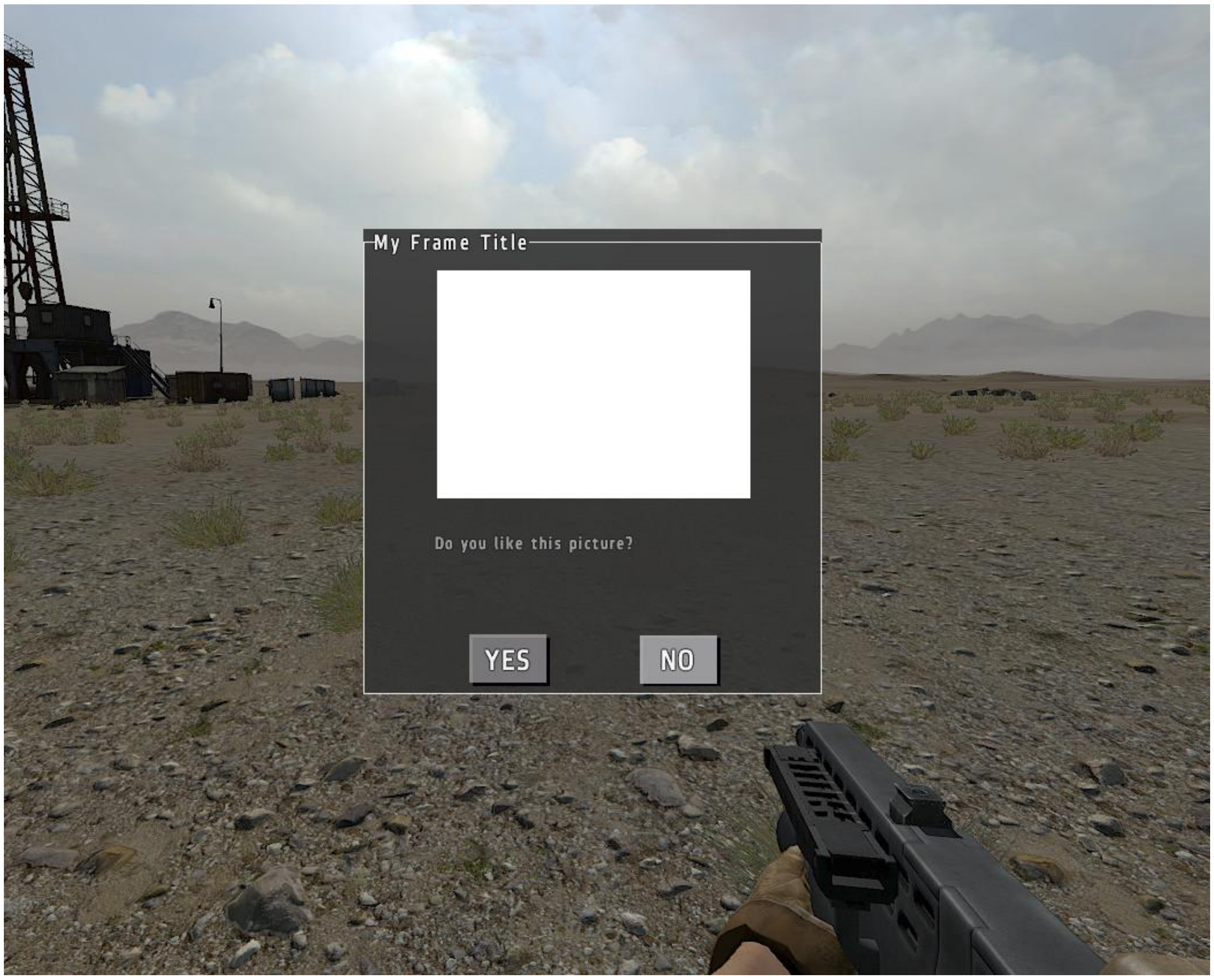
#include "defines.hpp"

#include "dialogs.hpp"

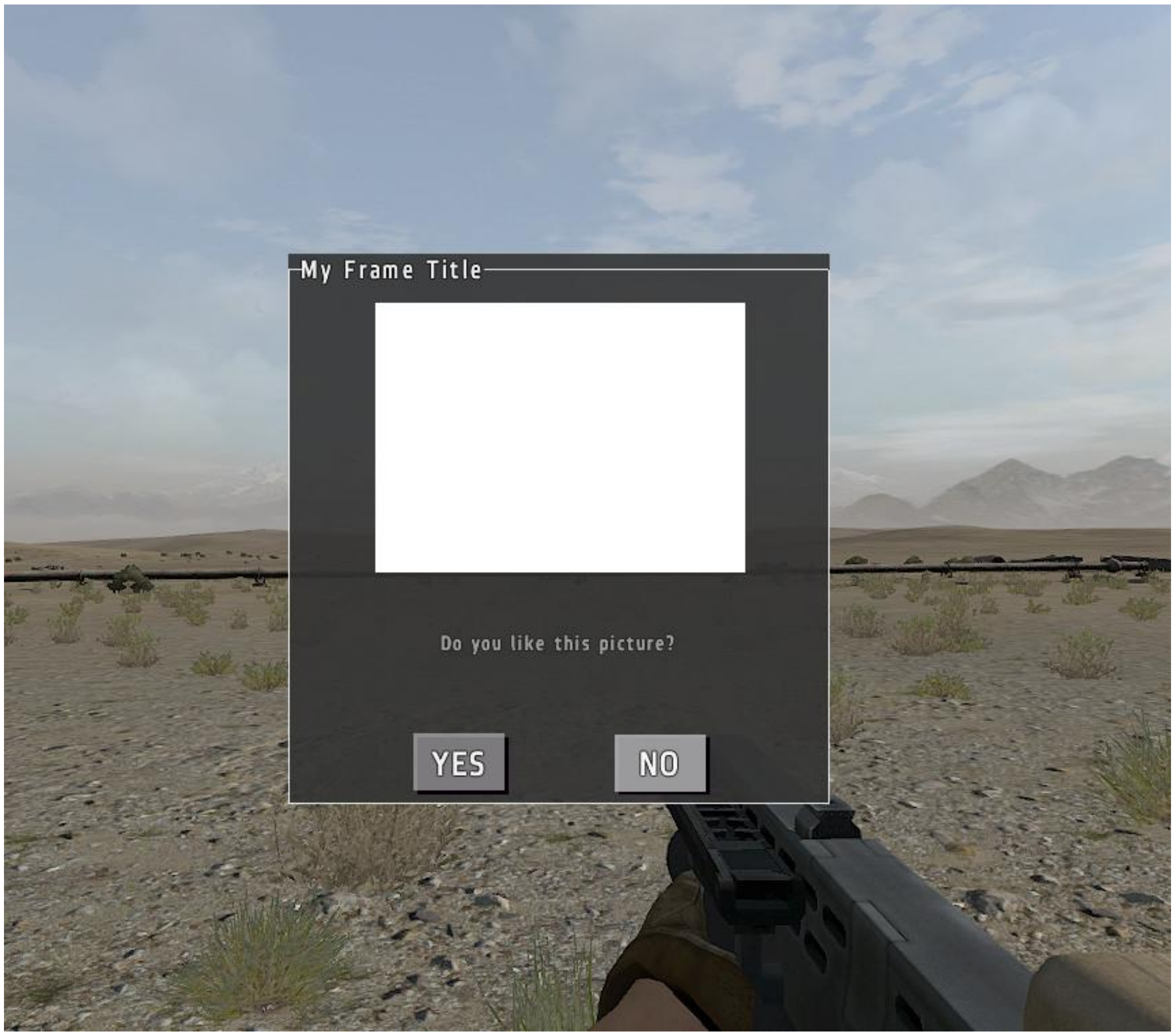
Save & close the description.ext. Go into the editor (**NOT THE GUI EDITOR**) and load your mission. You know the folder you created earlier, with the blank files?(Which should be blank no longer). Create a radio alpha trigger and put **_handle = CreateDialog "ICE_DIALOG"** into the OnAct. Ofcourse use whatever you named your dialog instead of ICE_DIALOG. It's pretty much just like calling a function if you know what that is, or how that works. If not then nvm. Anyhow,   preview the mission and check out your new dialog (radio trigger remember?).

<u>NOTE: WE'LL WORRY ABOUT UI COLORS AND ALL OF THAT JAZZ AFTER WE GET A WORKING DIALOG.</u>

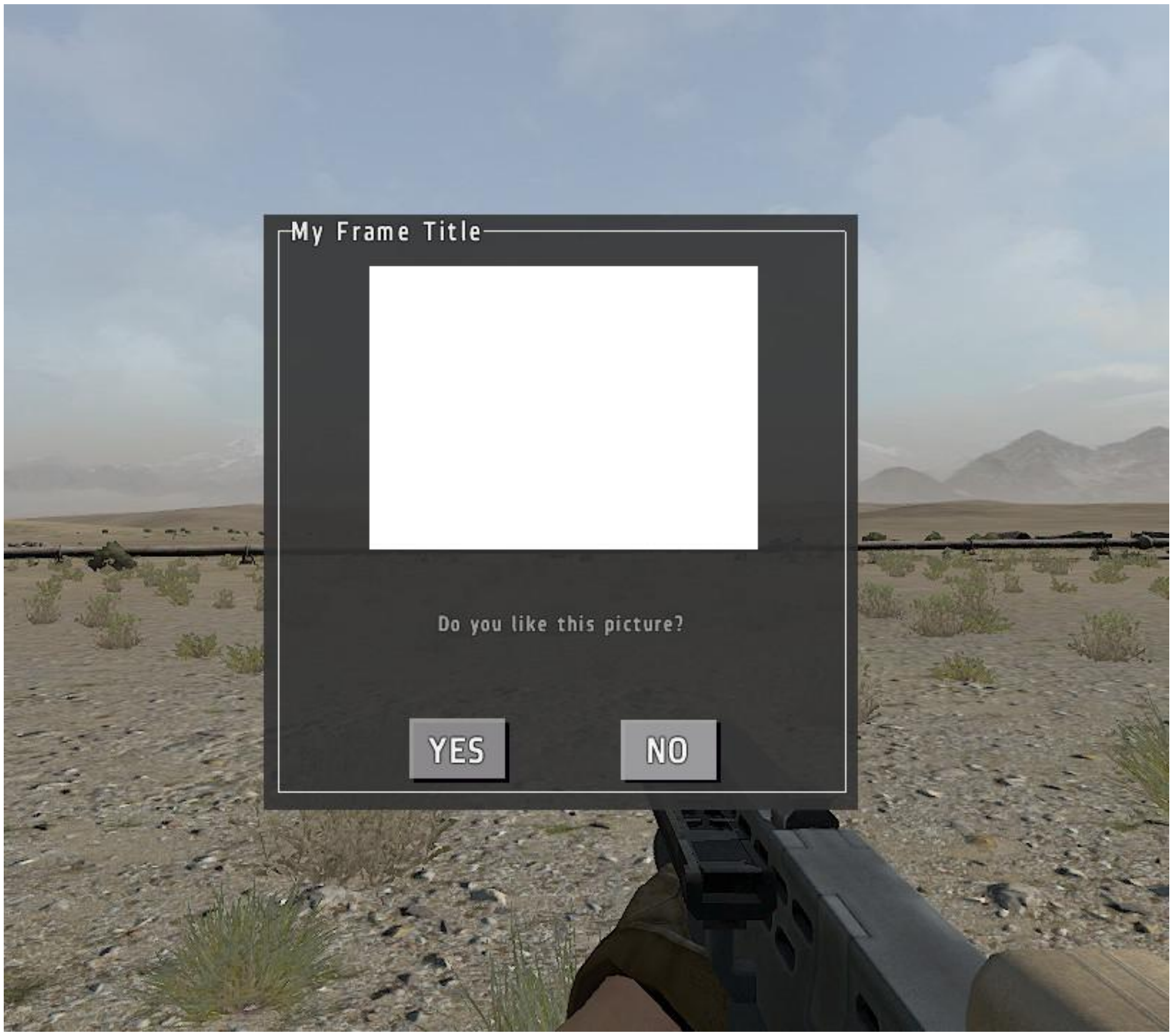
You should now have something like this when your activate the radio trigger:



The frame doesn't look right. The text is off center. You don't like what it says. There's no picture and you don't like the color scheme. We'll fix all of that. Open up your defines.hpp. Scroll down to the class rscText . Notice the style is set to style ST_MULTI or better to say 0x10. 0x10 is a hexadecimal for a normal number, I won't go into hexadecimals since I honestly just don't get it. But that's just another instance of wiki contributors using l33t code, when instead he or she could have simply used a normal value/number. You could infact go and replace all of the hexadecimals defines with the correct, normal number that the hexadecimal represents. Anyhow, since the name ST_MULTI can be defined as w/e you wish. It could be ST_BULLSHIT. It's the value (0x10) that matters. It's just good practice for the user to define the value (in this case 0x10) with a name that can easily identify what the style does. In this case, 0x10 or *"ST_MULTI"* as it were, is used to define multiple lines for text. So the text will wrap into the dialog. In this case, if you wanted to wrap a bunch of text (create a paragraph) into a dialog, you can do so because it will wrap instead of going beyond the borders of the dialog. In this case, we don't want style 0x10. We want ST_CENTER for the style or better known as 0x02. Go ahead and replace ST_MULTI with ST_CENTER. This way, the text will stay nice and centered. **Note: when assigning a style to a class, you can use the name or the value.** You should now have something like the image below. Notice the centered text.

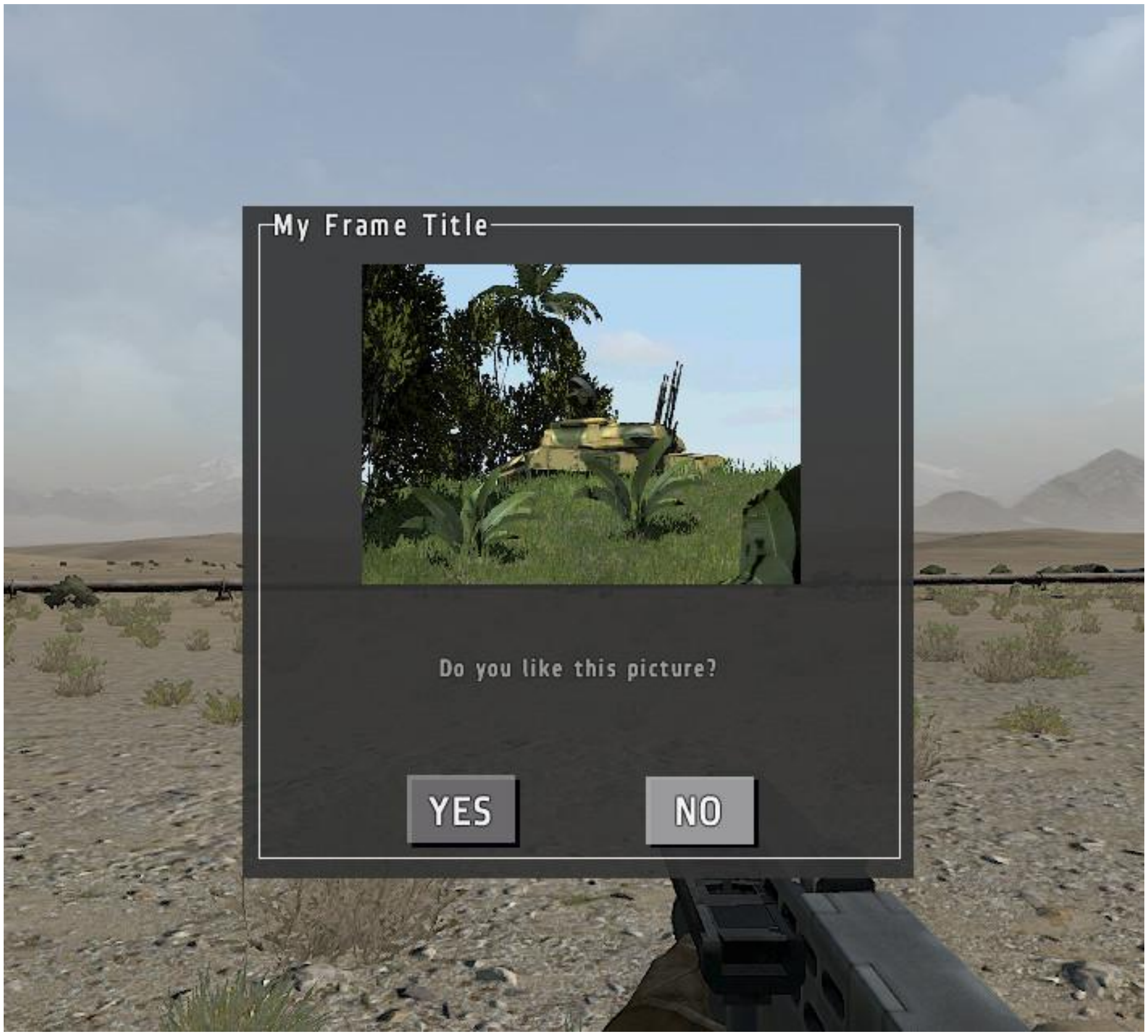


Now we need to make the background (BOX) slightly bigger than the frame, so it doesn't look absolutely terrible. Open your dialogs.hpp. We're going to edit our box class W: & H: (width & height) .You'll want to slightly increase the width and height values by the same number, so it stays uniform. IE; If you enlarge the height of the Background by X amount, then enlarge the width the same amount. You'll also need to adjust the X: value accordingly , as the BG will now be slightly offset. You'll need to recenter it (left to right = X value). This may take awhile to get aligned perfectly. Afterwords, you You should end up with something like:



So now we need to insert a picture, change the color scheme & make the buttons do stuff. Will start with the picture. Place your .paa (picture) in your mission folder. Make sure that your .paa (picture) is **roughly** the same size as the white picture window you see above. It doesn't have to be exact, but if it's too small or too big, it will look like shit when displayed. Okay, so to add your picture, open up your dialogs.hpp. Go to the Picture control/class. IE; ICE_PICTURE.

Replace **text = "#(argb,8,8,3)color(1,1,1,1)";** With >> **text = "yourPicture.paa";** Simple as that. When you preview the dialog you should now have something like:



Okay so now we'll change the color scheme. The BG, button color and text color & size. To do that, open up your defines.hpp. Each base class has some color properties. Text color, background color etc etc.

{1,1,1,1} == First 3 elements (numbers) make up the color. The color range is from 0 to 1. So like

{0.3, 0.6, 0.9,1}. You'll have to experiment to achieve the desired color. The last value is transparency. Again, ranging from 0 to 1. Experiment with these settings to make your UI look how you want.

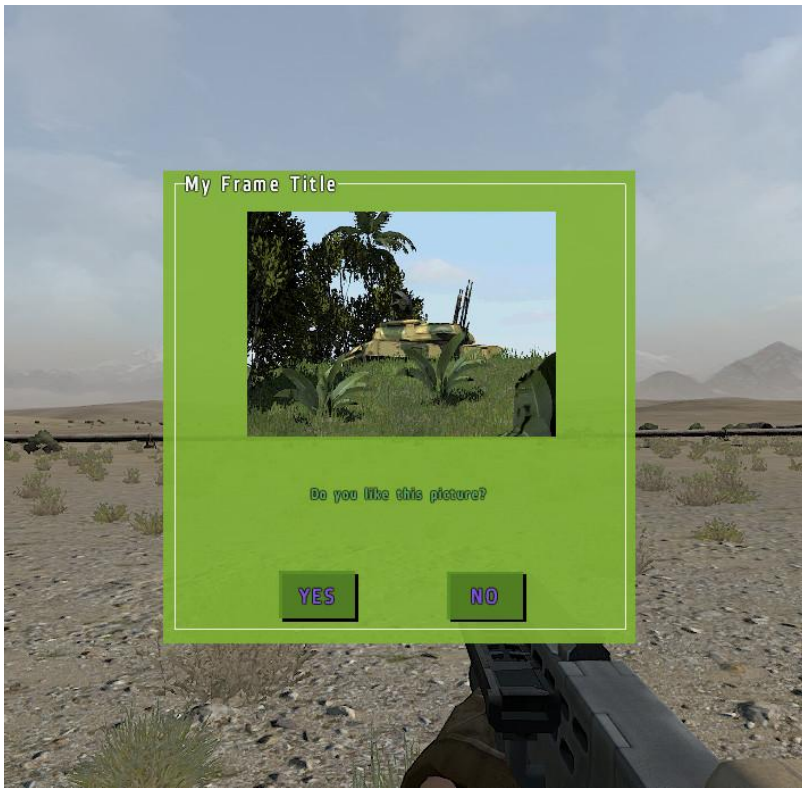
To change the font of some text, simply add the desired font under the font property in any base class. The available font types for Arma2 dialogs are listed below.

LucidaConsoleB
Zeppelin33
Zeppelin33Italic
Zeppelin32
EtelkaNarrowMediumPro
Bitstream
TahomaB
EtelkaMonospaceProBold

You can simply change the size of the text by setting the sizeEx property of any given class to the desired value. IE; **SizeEx = 0.02100;** Note:This value/percentage is small because it's relevant to the size of the screen. Be careful with this value, or I promise some unwanted results.

You will also notice that some styles, such as the button class, have unique properties that can be added to its base class. These properties pertain to the button itself ofcourse. Such as the different colors based on the buttons state. I'll let you experiment with those to come up with your own color scheme. Also, you'll notice the button can play sounds. Such as when it's clicked or mouse overed. Again, I'll let you determine the sounds you want. Simply replace the sound file with yours. But TBH, I really dig the button sounds in this tutorial.

When it's all said and done, you could only hope to have something as awesome as the image below:



But still, we have to make the buttons do stuff. In the case of this tutorial the buttons will run a script. Gohead and create 2 blank .sqf files in your mission folder. For the purpose of the tutorial we'll name the files hello1 & hello2 respectively. In hello1.sqf put **Hint "You like the picture";** In hello2.sqf put **Hint "You don't like the picture";**

Make sure both files are saved & good to go. Now open up your dialogs.hpp. Go to your YESBUTTON control/class and at the bottom put

action = "closeDialog 0;_nil=[]ExecVM ""hello1.sqf"""; (see below)

EXAMPLE SNIPPET:

```
class ICE_BUTTONYES: RscButton

    {

        idc = -1;

        text = "YES";

        x = 0.445239 * safezoneW + safezoneX;

        y = 0.625713 * safezoneH + safezoneY;

        w = 0.0398809 * safezoneW;

        h = 0.0404761 * safezoneH;

        action = "closeDialog 0;_nil=[]ExecVM ""hello1.sqf""";

    };
```

Do the same for your NOBUTTON. Except execVM hello2.sqf instead**. IMPORTANT!!!: Notice all of the quotes. It has to be exactly like this when you're running a script. You can get around this by simply precompiling a function and calling or spawning a function instead. IE; action = "closeDialog 0;_nil=[]Spawn YOUR_FNC";**

To keep within the guidelines of this guide, we'll just use ExecVM with the extra quotes for simplicity. Save & close your dialogs.hpp, go into the editor and save your work. Preview the mission. When you now hit either of the buttons, you should get a hint.

That concludes this tutorial. If you have any feedback, please direct it to the *"Dialog Tutorial For Noobs"* thread on the BI forums, editing & scripting section.