

Top of the class

Custom Cards

Summary

This is a component proposal that allows the player to pick a preset of a different card or use that 1 card and modify some values such as color, font, image and so on.

What this component offers:


- **CreateCard**: You can create a new card, all the buttons, texts, images can be created by the user.
- **CreateImage**: allows the user to create a new image, adding their own style and placing it on the card.
- **CreateText**: allows the user to customize the text
- **CreateAnimation**: allows the user to pick an image they want and select few of the made animations and use them how they like.
- **CreateButton**: allows the user to create a button and customize it how it looks and what it may do.
- **Setup Timer**: allows the user to load their own sound and when it should be triggered.

Link to Figma


<https://www.figma.com/file/wCMBc8IM5NAGa48qK6CeW0/Card-Pro?type=design&node-id=0%3A1&mode=design&t=4ZASVYUhdM9Jzscg-1>

References

Some images that may resemble or have some elements



Wandering Troll










Wandering Troll

Provide
Wandering Troll adds to Resources to army based on map square occupied

A troll is a being in Scandinavian folklore, including Norse mythology. In Old Norse sources, beings described as trolls dwell in isolated rocks, mountains, or caves, live together in small family units, and are rarely helpful to human beings.

In later Scandinavian folklore, trolls became beings in their own right, where they live far from human habitation, are not Christianized, and are considered dangerous to human beings. Depending on the source, their appearance varies greatly; trolls may be ugly.



	20 Attack
	20 Defense
	100 Health
	20 Speed
	40 Wealth
	30 Attraction



Features for inclusion

Timer: Example shown in Figma, there are few different types of timers,

1. A more normal timer where you can see the timer go down, when it goes down the color on that timer will become more red.
2. Themed - As you can see for the halloween part there is a pumpkin, everytime the pumpkin closes its mouth and then opens the countdown drops, it looks like it's eating the time.

Card answering questions: When answering questions there is an indication of getting it right or wrong. So like when getting it correct the card turns green, red if it's wrong.

1. The correct answer will turn the button green, red if wrong depending if the user clicked the correct one, if the timer runs out before the user clicks anything it will show the correct answer letting the person know what should've been the correct one.

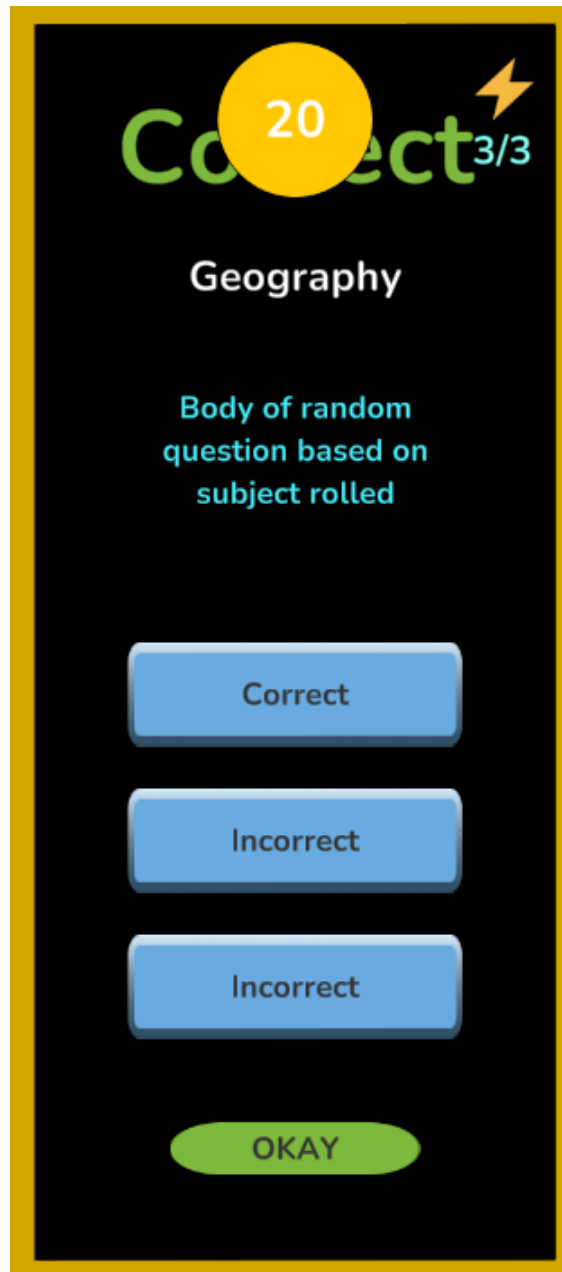
Themes: When Christmas, Easter or Halloween arrives you will get a custom card that is themed around that event.

1. The card has a simple different layout than other cards, it may have extra images or texts around the card that none other do also the color palette may be different as well.

Parameters for the card:

These are the parameters for the card that the user should be able to customize how they would like to see it as.

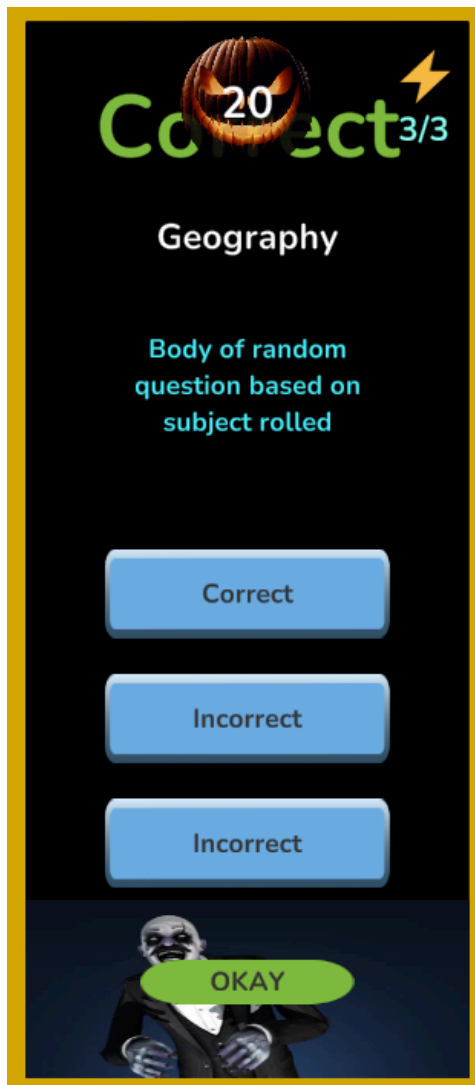
Default card appearance



- **Color (Background Color):** Where the Black color is on the card that's the background
- **Font:** What font type would the player like to use for any of the texts on the card.
- **Text 1 (Category/Subject text):** Takes the category/subject name from the game.

- **Text 2 (Question text):** Question text.
- **Text 3 (Answer 1 text):** The text parameter for the first answer
- **Text 4 (Answer 2 text):** The text parameter for the second answer
- **Text 5 (Answer 3 text):** The text parameter for the third answer
- **Text 6 (Timer text):** The text parameter for the timer
- **UI shape timer:** the UI parameter for the timer so you can change the shape to be a circle, rectangle, triangle or your own image.
- **UI timer color:** the color of the shape.
- **UI for Answers:** the shape of the answers UI, as you can see the 3 blue answers is a rectangle shape with curves, so you can change that shape into what the player would like to have.
- **UI for Answers color:** the colors on the answers UI shape.
- **Switch Subject:** The switch subject has parameters to change the UI image and the text color and font.
- **Okay button:** okay button the green button you see at the bottom of the card you can change the UI image and the text color and font on that object.

Advance parameters - Parameters that is possibly not for default cards



- **Animation:** You can create different animations, so adding your image and then picking what style the user would like. Going in a circle, vertical, horizontal, jumpscare, zig-zag which goes up and down.
- **Layout:** the user can select different layouts of cards to have a different style.
- **Create Image:** the user can create a new image inside the card, they can place their own image, color, x,y positions, width/height and material to add as optional.
- **Create Text:** the user can create their own text inside the card, with multiple options to pick from the text's position, size, rotation, input, font, bold or italic.

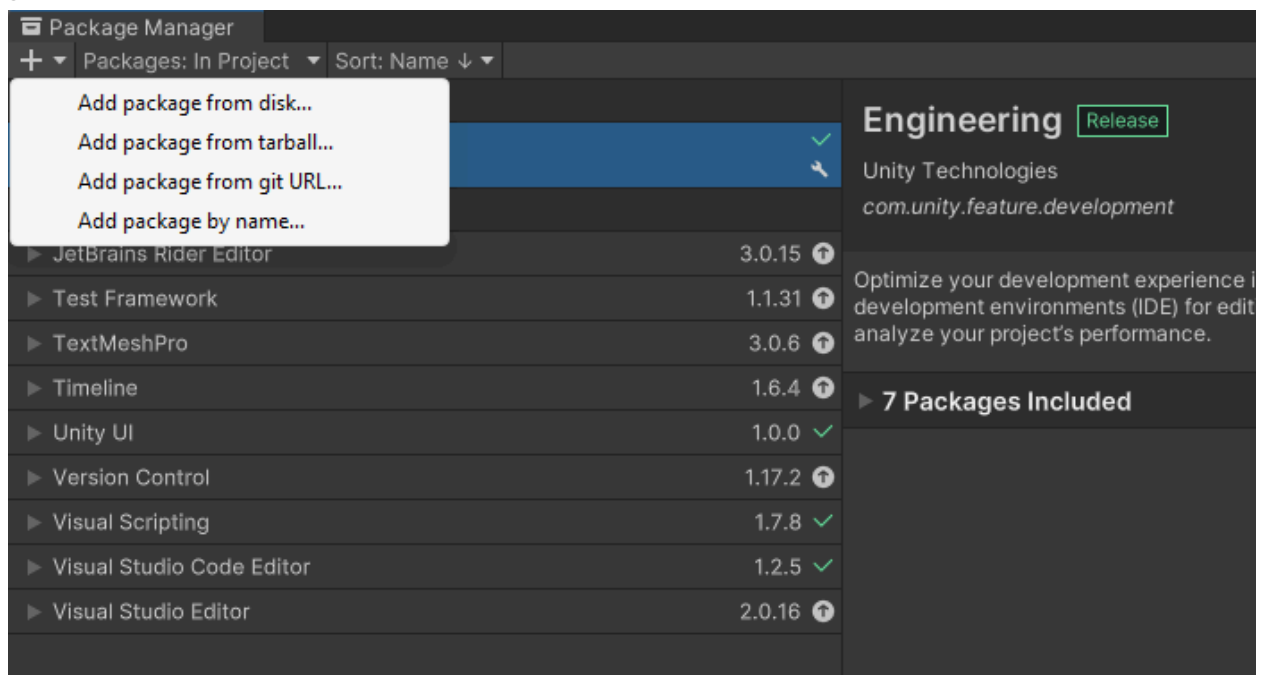
Guide

This is a guide to show how to install and use the component fully.

Step 1: How to install the component

Inside your project you need to go to windows -> Packet Manager

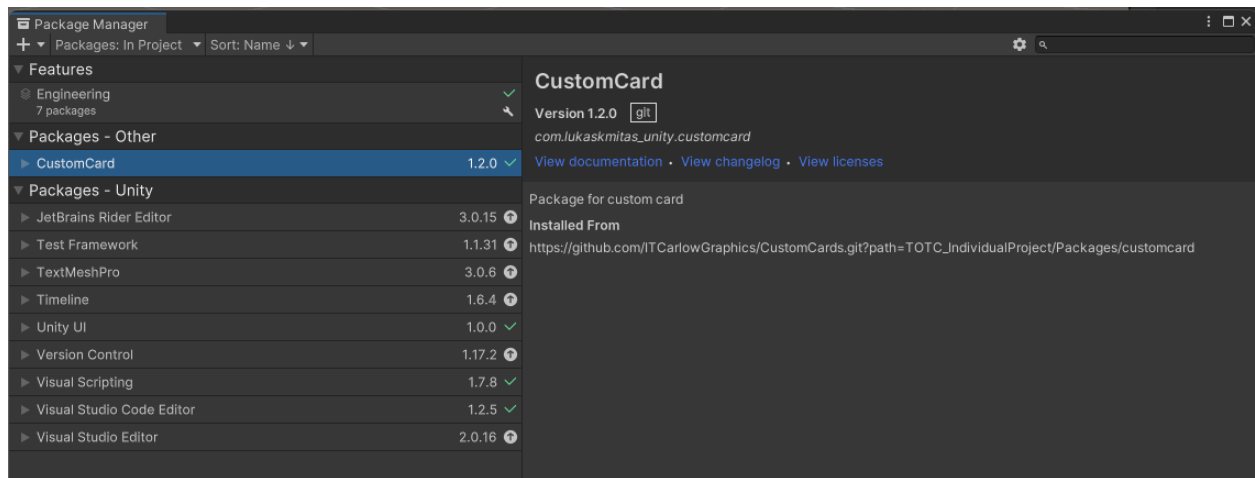
After clicking that a window will appear at the top left corner there is a plus sign + clicking that opens a tab, you need to accept the “Add package from git URL



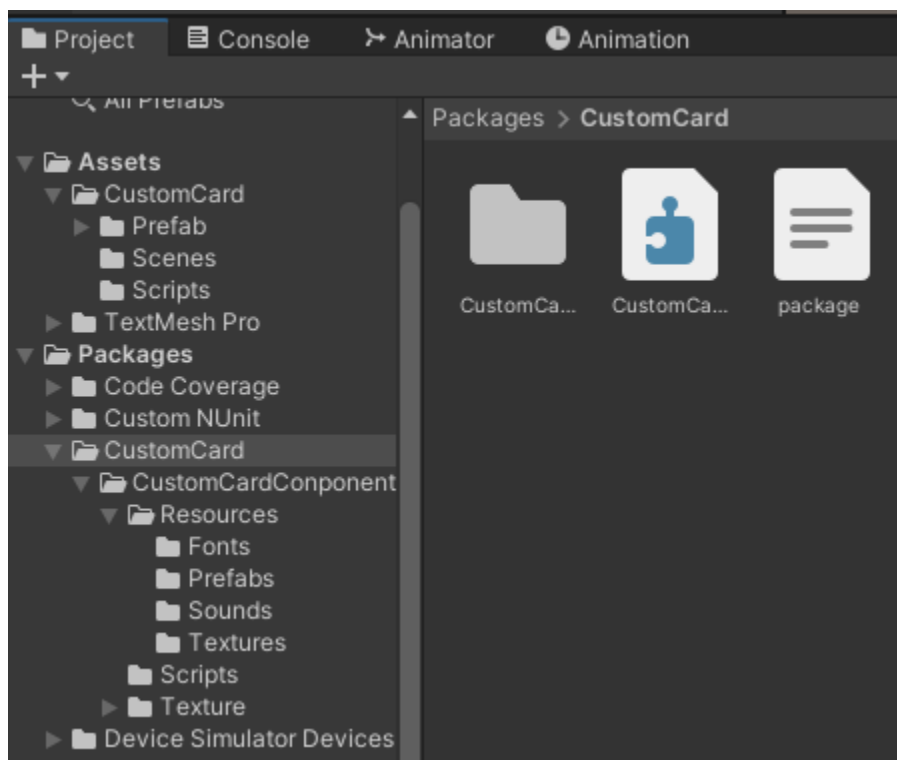
You need to paste the link below into the project to install the package from github

https://github.com/ITCarlowGraphics/CustomCards.git?path=TOTC_IndividualProject/Packages/customcard

After successfully installing it should look like this.

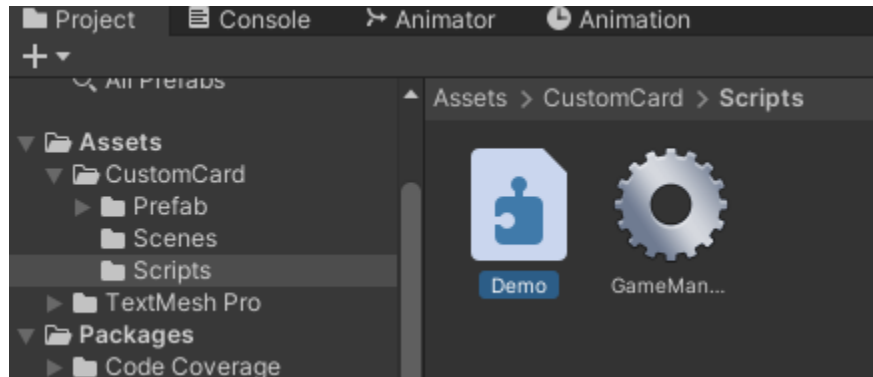


It shows the latest version, you will see the package installed with all the files in your project section inside the packages below you will see the new package that will appear there called “CustomCard” with all the assets like Fonts, Prefabs, Sounds, Textures and scripts.

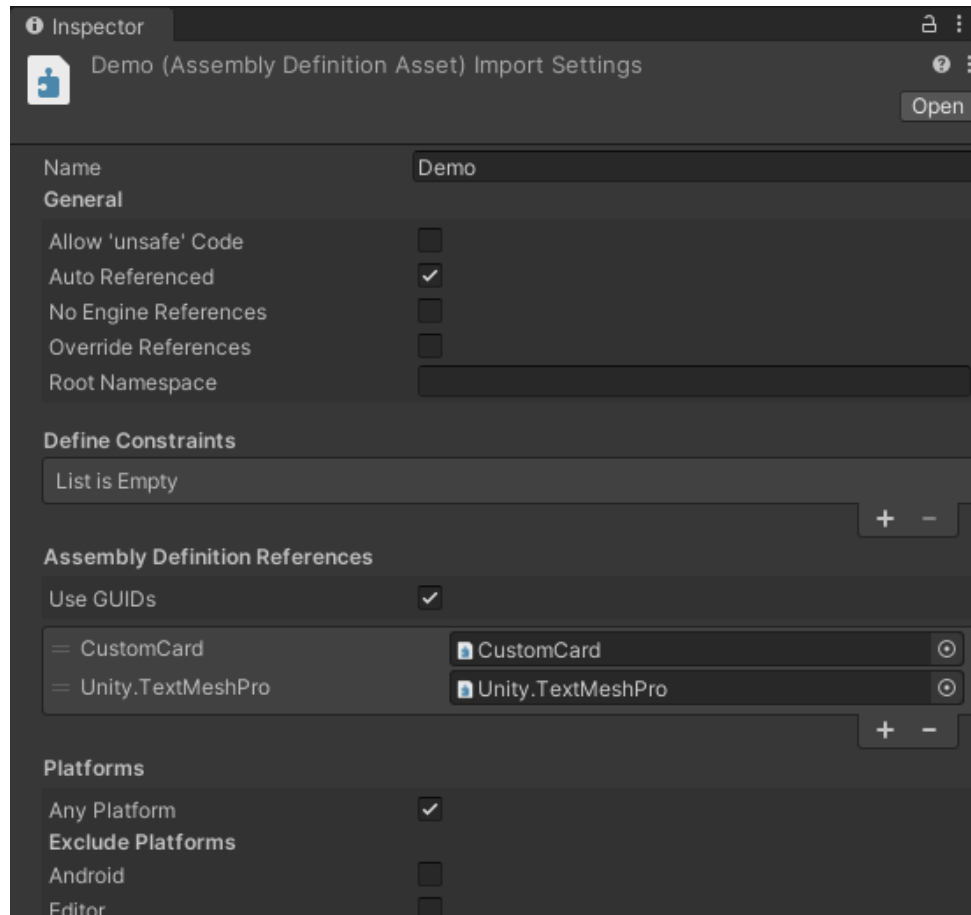


Step 2: How to use it, explanation about the component

This is an explanation on the component, everything that it can offer.
How to use it in your own component, to start off you will need to create a assembly definition



And inside the inspector you should attach the “CustomCard” this will allow the user to use the scripts freely without attaching any script into different objects and so on.



Inside your script where you want to instantiate it so for this example i'll use the Game manager.

From this component you can create multiple things such as instantiating the custom prefab, creating a custom image, text or animation, you can add them inside the prefab “Question card” and you can configure some settings with the timer component.

```
void Start()
{
    InstantiatePrefabInCanvas();

    CreateCustomImage();
    CreateCustomText();

    CreateAnimation();

    SetupTimerSound();
}
```

So to start off with instantiating the card inside your canvas you will always use the `Instantiate_Card` to call the script and since the script is an instance which allows you to use it anywhere. After typing an instance there will be multiple options i will show you the first one which is `createCard` this will give a lot of parameters where you can edit to your liking.

```
private void InstantiatePrefabInCanvas()
{
    // Current cards available

    // ##### DEFAULT CARD #####
    // CC_Default_Question Card

    // ##### THEMED CARDS #####
    // H_Theme_Question Card

    // Can choose between 2 colors RGB and HEX code, demonstration below it needs to be that format
    // RGB,255,100,255,
    // #1949c2
    // Fonts - Nunito Bold, AIW

    Instantiate_Card.Instance.createCard(
        "CC_Default_Question Card", // Name of preset card
        "1", // Layout
        "RGB,0,0,0,", // Background color
        "ffffff", "Nunito Bold", // Subject text
        "#4CF4FF", "Nunito Bold", // Question text
        "#7FB93D", "Nunito Bold", // Correct text
        "AnswerButton", "AnswerButton", "AnswerButton", // Answer button image
        "#38FF00", "#FF0000", "#FF0000", // Answer button highlight color
        "#323232", "#323232", "#323232", // Answer button text color
        "Nunito Bold", "Nunito Bold", "Nunito Bold", // Answer button font
        "Ellipse 12", "FFFFFF", "Nunito Bold", // Timer properties
        "Rectangle_383", "#323232", "Nunito Bold", // okay button prperties
        "Vector", "#88FFF1", "Nunito Bold" // switch subject properties
    );

    //Instantiate_Card.Instance.createCard("CC_Default_Question Card", "1", "RGB,0,0,0,", "ffffff", "
```

Here is a description of all parameters

```
Instantiate_Card.Instance.createCard()
void Instantiate_Card.createCard(string m_cardName, string layoutMode, string m_backgroundColor, string m_subjectTextColor,
    string m_subjectFontName, string m_questionTextColor, string m_questionTextFontName, string m_correctTextColor,
    string m_correctFontName, string m_button1ImageName, string m_button2ImageName, string m_button3ImageName,
    string m_button1HighlightColor, string m_button2HighlightColor, string m_button3HighlightColor, string m_button1TextColor,
    string m_button2TextColor, string m_button3TextColor, string m_button1FontName, string m_button2FontName,
    string m_button3FontName, string m_timerImageName, string m_timerTextColor, string m_timerTextFont, string m_okayImageName,
    string m_okayTextColor, string m_okayTextFont, string m_switchSubjectImageName, string m_switchSubjectTextColor,
    string m_switchSubjectFontName)
```

Since these are all strings you need to make sure the naming of the text is correct.

1. **m_cardName**: this will take the prefab from resources folder to use that as a template to edit other options inside it, you need to type the name of the prefab name.
2. **layoutMode**: this gives multiple different layouts of the card where the positions of the texts/images are so it won't look the same, you need to type either 1-3.
3. **m_backgroundColor**: you can type either the RGB or HEX code to get the color of the background of the card, the format has to be like this, for RGB "RGB,120,255,60," and for HEX code its "#26D155" this applies to all colors
4. The following parameters: **m_subjectTextColor**, **m_subjectFontName**, these allow you to change the color and the font of the object called "QuestionSubject".
5. The following parameters: **m_questionTextColor**, **m_questionTextFontName**, these allow you to change the color and font of the object called "QuestionText"
6. The parameters: **m_correctTextColor** and **m_correctFontName**, these allow you to change the color and font of the object called "CorrectText".
7. The parameters: **m_button1ImageName**, **m_button2ImageName** and **m_button3ImageName**, these allow you to change the UI image source for the answer buttons there are 3 answer buttons.
8. The parameters: **m_button1HighlightColor**, **m_button2HighlightColor** and **m_button3HighlightColor**, These allow you to change the highlight color of the answer buttons so this is the color that appears when pressing the button.
9. The parameters: **m_button1TextColor**, **m_button2TextColor** and **m_button3TextColor**, these allow you to change the color of the text inside the buttons
10. The parameters: **m_button1FontName**, **m_button2FontName** and **m_button3FontName**, these allow you to change the font of the text of the answer buttons.
11. The parameters: **m_timerImageName**, **m_timerTextColor** and **m_timerTextFont**, these allow you to change the properties of the Timer object such as the UI image source, the text color and font.

12. The parameters: **m_okayImageName**, **m_okayTextColor** and **m_okayTextFont**, these allow you to change the okay button properties such as the UI image source, text color and font.
13. The parameters: **m_switchSubjectImageName**, **m_switchSubjectTextColor** and **m_switchSubjectFontName**, these allow you to change the properties of the object called "Switch Subject" you can change the UI image, text color and font.

All the user needs to do is type the function name `createCard` and type the configuration they want to have, that's all and call it in the start function or wherever they wish.

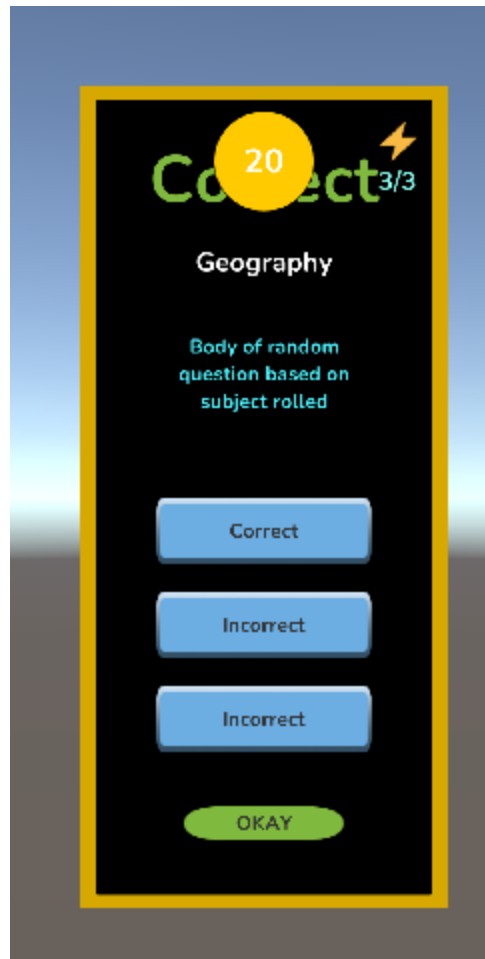
Now before moving into the other functions to create stuff I will show some examples of the `createCard` parameters.

Example 1

Now this here will be the default parameters

```
Instantiate_Card.Instance.createCard(
    "CC_Default_Question Card", // Name of preset card
    "1", // Layout
    "RGB,0,0,0,", // Background color
    "#ffffff", "Nunito Bold", // Subject text
    "#4CF4FF", "Nunito Bold", // Question text
    "#7FB93D", "Nunito Bold", // Correct text
    "AnswerButton", "AnswerButton", "AnswerButton", // Answer button image
    "#38FF00", "#FF0000", "#FF0000", // Answer button highlight color
    "#323232", "#323232", "#323232", // Answer button text color
    "Nunito Bold", "Nunito Bold", "Nunito Bold", // Answer button font
    "Ellipse 12", "#FFFFFF", "Nunito Bold", // Timer properties
    "Rectangle_383", "#323232", "Nunito Bold", // okay button prperties
    "Vector", "#88FF1", "Nunito Bold" // switch subject properties
);
```

And this is how it looks like when instantiated

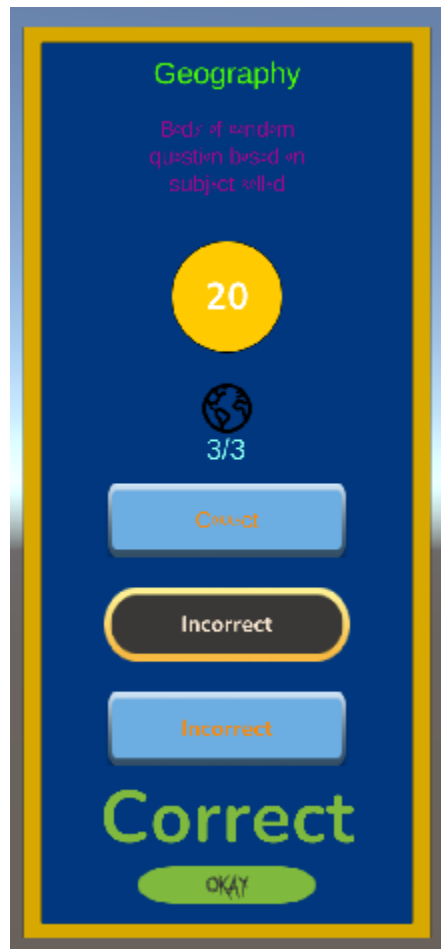


Example 2

Some parameter variables are changed

```
Instantiate_Card.Instance.createCard(
    "CC_Default_Question Card", // Name of preset card
    "2", // Layout
    "RGB,0,65,150,", // Background color
    "#4dff00", "SCP", // Subject text
    "#a3088c", "AIW", // Question text
    "#7FB93D", "Nunito Bold", // Correct text
    "AnswerButton", "Rectangle 424", "AnswerButton", // Answer button image
    "#38FF00", "#FF0000", "#FF0000", // Answer button highlight color
    "#ff8800", "#fcead4", "#ff8800", // Answer button text color
    "AIW", "Nunito Bold", "Nunito Bold", // Answer button font
    "Ellipse 12", "#FFFFFF", "Nunito Bold", // Timer properties
    "Rectangle_383", "#323232", "Nightcore", // okay button properties
    "earth", "#88FFF1", "AIW" // switch subject properties
);
```

This is how it looks like when its instantiated with these new edited parameter variables



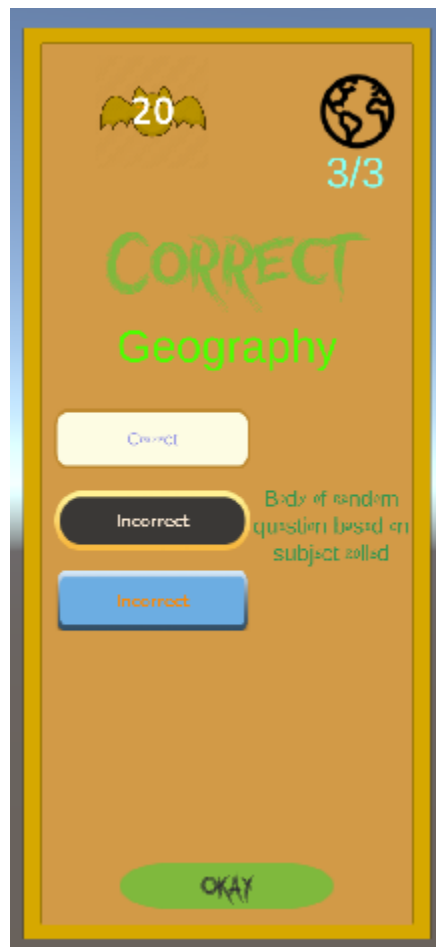
Example 3

Last example for this part

```

Instantiate_Card.Instance.createCard(
    "CC_Default_Question_Card", // Name of preset card
    "3", // Layout
    "#fab855", // Background color
    "#4dffb0", "SCP", // Subject text
    "RGB,50,150,80,", "AIW", // Question text
    "#7FB93D", "Nightcore", // Correct text
    "Rectangle 429", "Rectangle 424", "AnswerButton", // Answer button image
    "#38FF00", "#FF0000", "#FF0000", // Answer button highlight color
    "RGB,150,150,230,", "#fced4", "#ff8800", // Answer button text color
    "AIW", "Nunito Bold", "Nunito Bold", // Answer button font
    "Bat", "#FFFFFF", "Nunito Bold", // Timer properties
    "Rectangle_383", "#323232", "Nightcore", // okay button prperties
    "earth", "#88FF1", "AIW" // switch subject properties
);

```



This is what the createCard does: it allows the user to edit the variables that are available in the createCard.

Next we have is the **createImage** function from the “Instantiate_Card” script

```
Custom_Card.Instance.createImage(  
    "Canvas", // parent name  
    "Custom Image With Behaviors", // image name  
    "0", "400", // positions  
    "800", "350", // width and height  
    "0", // rotation  
    "MadPenguin", // image source  
    "#FFFFFF", // image color  
    "", // material  
    true, // Attach ImageBehaviours script  
    true, // Enable image swapping  
    "Ellipse13", "Ellipse14", 1.0f, // Image swap parameters  
    true, // Enable color changing  
    "#FFFFFF", "#FF0000", 2.0f // Color change parameters  
);
```

This will create a new image inside the “Question card” object

```
public void createImage(  
    string parentName, // parent name  
    string imageName, // image name  
    string xPosition, string yPosition, // x,y positions  
    string width, string height, // width and height of the image  
    string rotation, // rotation  
    string imageSource, // image source  
    string imageColor, // color of image  
    string imageMaterial, // material to use  
    bool attachImageBehaviours = false, // to use the script  
    bool enableImageSwap = false, // enable image swapping  
    string image1Name = null, string image2Name = null, float swapInterval = 0.5f, // properties for swap image behaviour  
    bool enableColorChange = false, // enable color changing  
    string startColor = null, string endColor = null, float colorChangeDuration = 1.0f // properties for color changing  
)
```

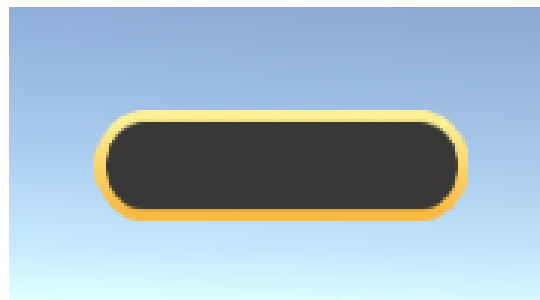
There are the parameters for creating a new image

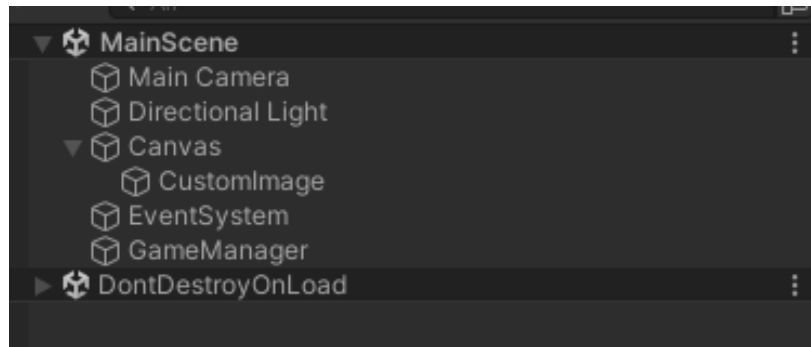
1. **parentName**: This is used to create the card inside the “Question Card” and not be somewhere else in the canvas that can obstruct other game objects.
2. **imageName**: this allows the user to create the name for that object
3. **xPosition** and **yPosition**: this initializes the positions for that image
4. **Width** and **height**: sets the width and height of the object.
5. **Rotation**: sets the rotation of the image object
6. **imageSource**: gets the source of the image
7. **imageColor**: sets the color for that image

8. **ImageMaterial**: sets the material for that image, you can leave the string empty if the user doesn't want to use it.
9. **attachImageBehaviours**: this gives an option for the user if they would like to attach a script onto the object to create different stuff
10. **enableImageSwap**: to allow the user to swap between 3 images.
11. **image1Name, image2Name, swapInterval**: the image names are to get the source image and the swapInterval is to check how fast you would like to swap the images in between.
12. **enableColorChange**: as the name suggests, it allows you to change color on the image.
13. **startColor, endColor, colorChangeDuration**: start and end color types and the colorChangeDuration is how long you would want the color to remain.

Here is some example of me creating the image on a empty canvas
Example 1

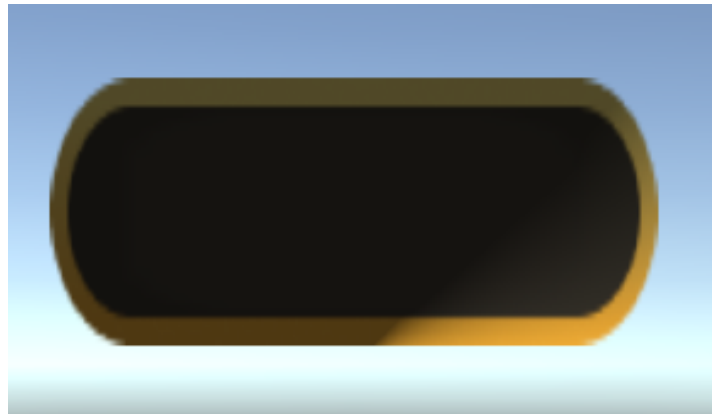
```
Instantiate_Card.Instance.createImage(  
    "Canvas", // Parent name  
    "CustomImage", // Image name  
    "0", "200", // Position  
    "400", "120", // Size  
    "0", // Rotation  
    "Rectangle 424", // Image source  
    "#FFFFFF", // Image color  
    "" // Image material (optional)  
);
```





Example 2

```
Instantiate_Card.Instance.createImage(  
    "Canvas", // Parent name  
    "Image with material", // Image name  
    "0", "200", // Position  
    "800", "350", // Size  
    "0", // Rotation  
    "Rectangle 424", // Image source  
    "#FFFFFF", // Image color  
    "mat1" // Image material (optional)  
);
```



It's pretty easy to use this createCard and understand it,
Next we have the createText, same thing as the createImage it has nearly
the same parameters in the function.

```

Instantiate_Card.Instance.createText(
    "Question Card", // Parent name
    "CustomText", // Text name
    "100", "100", // Position
    "300", "300", // Size
    "30", // Rotation
    "THIS IS A CUSTOM TEXT!!!", // Text content
    "#F8BA42", // Text color
    "AIW", // Font name
    50, // Font size
    true, // Bold
    false // Italic
);

```

```

public void createText(
    string parentName, // parent gameobject
    string textName, // Name for the new text GameObject
    string xPosition, string yPosition, // X and Y positions
    string width, string height, // Width and height
    string rotation, // Rotation
    string textContent, // Text content
    string textColor, // Text color
    string fontName, // Font name
    int fontSize, // Font size
    bool isBold, // Bold text
    bool isItalic // Italic text
)

```

These are the parameters of this createText

1. **parentName**: where would the user like to create the card, on which parent.
2. **textName**: the name of this text object
3. **xPosition**, **yPosition**: these are to set the position of the text object
4. **Width** and **height**: setting the width and height of the text object.
5. **Rotation**: setting the rotation of the object.
6. **textContent**: what would the user like to put in the text.
7. **textColor**: setting the color
8. **fontName**: setting the font name
9. **fontSize**: setting the size of the text
10. **isBold**: setting if the text is bold or not by typing true or false.
11. **isItalic**: setting if the text is italic or not.

Example 1

```
Instantiate_Card.Instance.createText(  
    "Canvas", // Parent name  
    "CustomText", // Text name  
    "100", "100", // Position  
    "300", "300", // Size  
    "30", // Rotation  
    "THIS IS A CUSTOM TEXT!!!", // Text content  
    "#F8BA42", // Text color  
    "AIW", // Font name  
    80, // Font size  
    true, // Bold  
    true // Italic  
);
```

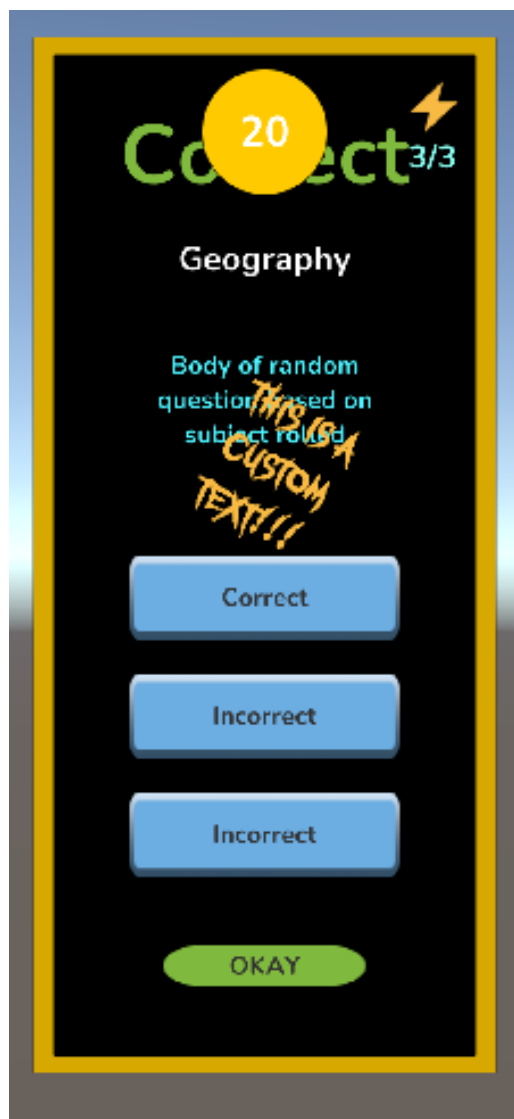


Example 2

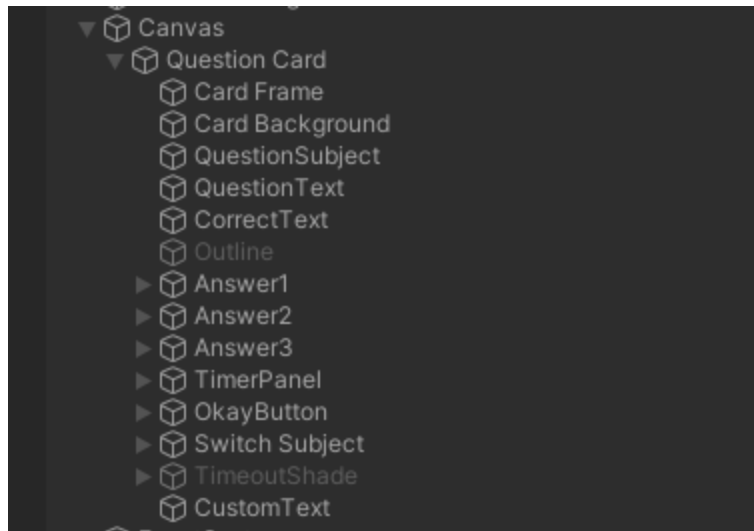
```

Instantiate_Card.Instance.createText(
    "Question Card", // Parent name
    "CustomText", // Text name
    "100", "100", // Position
    "300", "300", // Size
    "-30", // Rotation
    "THIS IS A CUSTOM TEXT!!!", // Text content
    "#F8BA42", // Text color
    "Nightcore", // Font name
    50, // Font size
    true, // Bold
    false // Italic
);

```



Here is setting inside the “Question card”



Next we have the **createAnimation** function for the Instantiate_Card

```
Instantiate_Card.Instance.createAnimation(  
    "Question Card", // Parent name  
    "zig zag animation", // animation name  
    "-350", "500", // Initial position  
    "100", "100", // Size  
    "0", // Rotation  
    "Bat", // Image source  
    "#FFFFFF", // Image color  
    3.0f, // Move speed  
    "zigzag", // Move pattern  
    1 // Quantity  
);
```

```
public void createAnimation(  
    string parentName, // parent GameObject  
    string animationName, // animation name  
    string xPosition, string yPosition, // Initial X and Y positions  
    string width, string height, // Width and height  
    string rotation, // Initial rotation  
    string imageSource, // Image source  
    string imageColor, // Image color  
    float moveSpeed, // Movement speed  
    string movePattern, // Movement pattern (e.g. "zigzag", "vertical", "horizontal", "jumpscare")  
    int quantity // Number of entities to create  
)
```

The parameters for this are:

1. **parentName**: where exactly do you want to place this object
2. **animationName**: the name of this object

3. **xPosition** and **yPosition**: the position of this game object
4. **Width** and **height**: how big the object is
5. **Rotation**: the rotation of the object.
6. **imageSource**: the image for the animation
7. **imageColor**: the color of the image
8. **moveSpeed**: the speed which the object should be moving
9. **movePattern**: In here you can choose from the options to have different type of effects for the object so there are the effects the you can have
 - **vertical**: it only moves in that direction
 - **horizontal**: moves the object into that direction
 - **zigzag**: makes the object move up and down in a loop
 - **circular**: Moves the object in a circular way in a loop
 - **scale**: it scales the object, makes it big and small in a loop
 - **jumpscare**: the object expands rapidly and then fades away, runs once
10. **Quantity**: how many of these object entities would the user like for that 1 function

Here is an example of all of them being used



I have some stuff to do with the timer function as well to help customize for the user making it easier for them.

```
CC_Timer.Instance.LoadSoundFromResources("beam-wowowfast");  
CC_Timer.Instance.playSoundAt = 8.0f;  
CC_Timer.Instance.TimerIsRunning(true);
```

So using the **LoadSoundFromResources** function this can load any sound you would like from resources folder

PlaySoundAt is to make the sound play at that specific time when it hits, there are many different sounds, and instead of editing the sound itself you can play the sound on a specific time instead.

TimerIsRunning you need to have this to run the time

Create button customization allows the user to create a button and customize a lot of stuff inside it.

```
Custom_Card.Instance.createButton(  
    "Canvas", // Parent GameObject name  
    "Custom Button", // Button name  
    "100", "100", // Position  
    "300", "100", // Size  
    "0", // Rotation  
    "Click Me to pop", // Button text  
    "#FFFFFF", // Text color  
    "Nunito Bold", // Font name  
    40, // Font size  
    true, // Bold  
    false, // Italic  
    "Rectangle 424", // Button image source  
    "#FF0000", // Highlight color  
    //buttonHandler.OnCustomButtonClick // OnClick action  
    () => {  
        GameObject button = GameObject.Find("Custom Button");  
        ButtonHandler buttonHandler = button.GetComponent<ButtonHandler>();  
        buttonHandler.OnCustomButtonClick();  
    } // OnClick action  
);
```

```
public void createButton(  
    string parentName, // name of parent  
    string buttonName, // name of button  
    string xPosition, string yPosition, // x,y positions  
    string width, string height, // width and height  
    string rotation, // rotation  
    string buttonText, // the buttons text  
    string textColor, // text color  
    string fontName, // text font  
    int fontSize, // text size  
    bool isBold, // text being bold  
    bool isItalic, // text being italic  
    string buttonImage, // button image source  
    string highlightColor, // highlight color  
    UnityEngine.Events.UnityAction onClickAction // Action to call on button click  
)
```

So the parameters for this are:

1. **parentName**: name of parent
2. **buttonName**: name of button

3. **xPosition, yPosition**: x,y positions
4. **width, height**: width and height
5. **rotation**: rotation
6. **buttonText**: the buttons text
7. **textColor**: text color
8. **fontName**: text font
9. **fontSize**: text size
10. **isBold**: text being bold
11. **isItalic**: text being italic
12. **buttonImage**: button image source
13. **highlightColor**: highlight color
14. **UnityEngine.Events.UnityAction onClickAction**: the function to make it do something, if its clicked on it will do that

Now most of these parameters are easy to understand but the last one I will go in further detail.

Now the button gets 2 scripts attached to it when its being created, the reason for this is to separate as much as possible so this code

```
() => {  
    GameObject button = GameObject.Find("Custom Button");  
    ButtonHandler buttonHandler = button.GetComponent<ButtonHandler>();  
    buttonHandler.OnCustomButtonClick();  
} // onClick action
```

It calls the function from another script to do something so what you need to is the find that object you are creating and get the ButtonHandler script from it and then use the OnCustomButtonClick a function that will scale the object up and rotate it slightly to make it look better when pressing the button.

```

public void OnCustomButtonClick()
{
    // Find the ButtonEffects component on the same GameObject and call its OnButtonClick method
    GameObject buttonGameObject = gameObject;
    Debug.Log(gameObject);

    ButtonEffects clickEffect = buttonGameObject.GetComponent<ButtonEffects>();

    if (clickEffect != null)
    {
        clickEffect.OnButtonClick();
    }
    else
    {
        Debug.LogError("ButtonEffects component not found on this GameObject!");
    }
}

```

OnButtonClick leads to another script which does the animation, IEnumerator stuff. So you can go about this 2 ways 1 is make the functions inside the ButtonHandler.

```

public class ButtonHandler : MonoBehaviour
{
    public void OnCustomButtonClick()
    {
        // Find the ButtonEffects component on the same GameObject and call its OnButtonClick method
        GameObject buttonGameObject = gameObject;
        Debug.Log(gameObject);

        ButtonEffects clickEffect = buttonGameObject.GetComponent<ButtonEffects>();

        if (clickEffect != null)
        {
            clickEffect.OnButtonClick();
        }
        else
        {
            Debug.LogError("ButtonEffects component not found on this GameObject!");
        }
    }

    public void onButtonClickCreateAnimation()
    {
        Debug.Log("ANIMATION APPEARS");

        Custom_Card.Instance.createAnimation(
            "Canvas", // Parent name
            "zig zag animation", // animation name
            "-350", "500", // Initial position
            "100", "100", // Size
            "0", // Rotation
            "Bat", // Image source
            "#FFFFFF", // Image color
            3.0f, // Move speed
            "zigzag", // Move pattern
            1 // Quantity
        );
    }
}

```

Or you can just make a function inside where you are creating the buttons etc. by doing this instead

```
Custom_Card.Instance.createButton(  
    "Canvas", // Parent GameObject name  
    "Custom Button 3", // Button name  
    "100", "-400", // Position  
    "400", "200", // Size  
    "0", // Rotation  
    "Click Me for create image", // Button text  
    "#FFFFFF", // Text color  
    "Nunito Bold", // Font name  
    40, // Font size  
    true, // Bold  
    false, // Italic  
    "Rectangle 424", // Button image source  
    "#FF0000", // Highlight color  
    //buttonHandler.onButtonClickCreateAnimation // OnClick action  
    CreateCustomImage  
);
```

Only call the function that you need, very easy you can do both ways first is cleaner though so the Game Manager script or any other will not be clustered and you can have more room then.

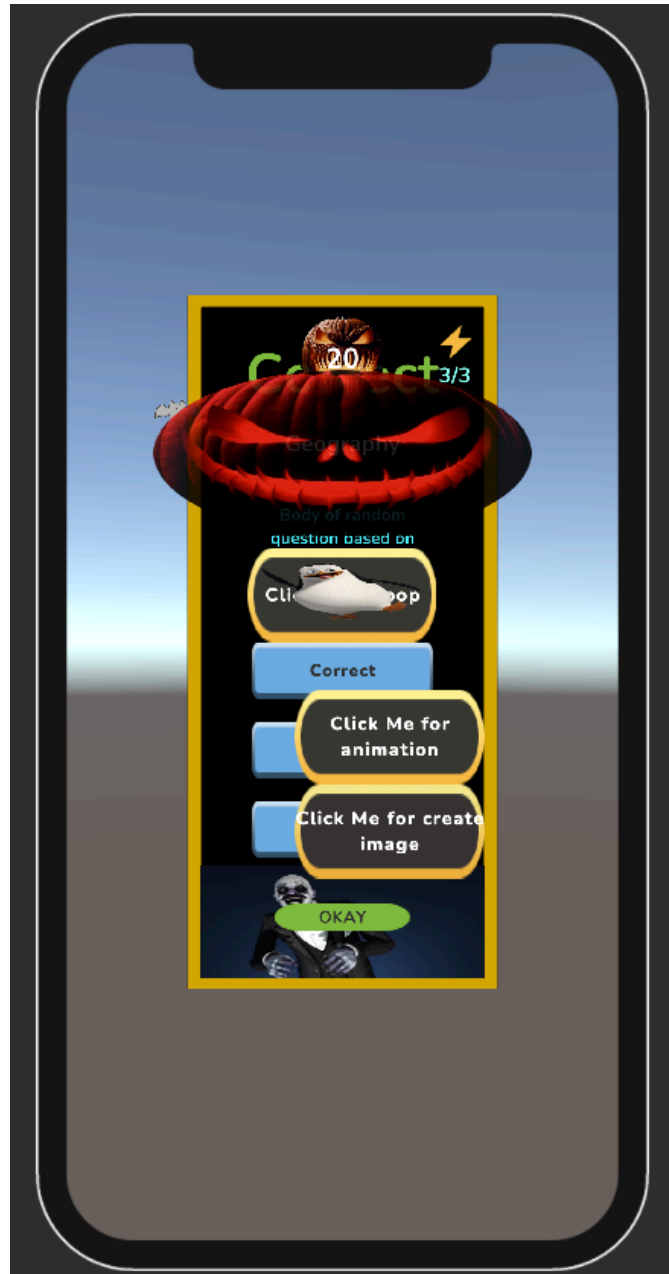
Here is some example of the 3 buttons code being run in unity.



Clicking on the “click me for animation” is to create the animation bat doing zigzag behavior



And clicking on the “Click me for image” created an image



All of these functions are being instantiated when the game is run such as the create image, text, animation, card itself and so on.

Summary

All you need to do to create new stuff is find where you want to place the code eg. Game Manager for this example, and the create a function in the start function like - `InstantiatePrefabInCanvas()`

Inside that function you make the createCard and so on, that's all like this examples shown below.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class GameManagerTest : MonoBehaviour
6  {
7
8      void Start()
9      {
10         InstantiatePrefabInCanvas();
11
12         //CreateCustomImage();
13         //CreateCustomText();
14         //CreateCustomButton();
15         //CreateAnimation();
16         //SetupTimerSound();
17     }
18
19     private void InstantiatePrefabInCanvas()
20     {
21         // Current cards available
22
23         // ##### DEFAULT CARD #####
24         // CC_Default_Question Card
25
26         // ##### THEMED CARDS #####
27         // H_Theme_Question Card
28
29         // Can choose between 2 colors RGB and HEX code, demonstration below it needs to be that format
30         // RGB,255,100,255,
31         // #1949c2
32         // Fonts - Nunito Bold, AIW, Nightcore
33         // Sound - 20-second-timer-v3, beam-wowowfast, moo-death
34
35         Custom_Card.Instance.createCard(
36             "CC_Default_Question Card", // Name of preset card
37             "1", // Layout
38             "RGB,0,0,0,", // Background color
39             "ffffff", "Nunito Bold", // Subject text
40             "#4CF4FF", "Nunito Bold", // Question text
41             "#7FB93D", "Nunito Bold", // Correct text
42             "AnswerButton", "AnswerButton", "AnswerButton", // Answer button image
43             "#38FF00", "#FF0000", "#FF0000", // Answer button highlight color
44             "#323232", "#323232", "#323232", // Answer button text color
45             "Nunito Bold", "Nunito Bold", "Nunito Bold", // Answer button font
46             "Ellipse 12", "FFFFFF", "Nunito Bold", // Timer properties
47             "Rectangle_383", "#323232", "Nunito Bold", // okay button properties
48             "Vector", "#88FFF1", "Nunito Bold" // switch subject properties
49         );
50     }
51
52     //Instantiate_Card.Instance.createCard("CC_Default_Question Card", "1", "RGB,0,0,0,", "ffffff", "Nunito B
53 }
54
```

This will create the whole functional card.