

ImpactLab - Maximizing your CS Skills

Lecture 6: Finishing Up VI

Summer 2022

School of Computing
and Data Science

Wentworth Institute of
Technology



Wentworth
Computing & Data Science

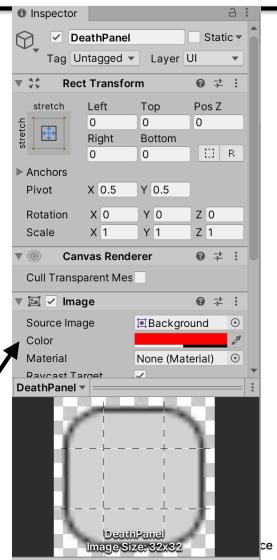
Game Over Screen

We're going to be using the standard Unity UI system to create our Main Menu and Game Over screen

Start by creating a new UI->Image in your Hierarchy.

This automatically creates a Canvas for you that contains the Image object as a child.

We'll have to set the image that will be displayed. Initially, we'll set the tint to a solid color (with transparency). I'm using the standard button image that is build in.



What's Left?

- Particle Effects ✓
- Animation ✓
- Start/Game Over Screen
- Sound/Music

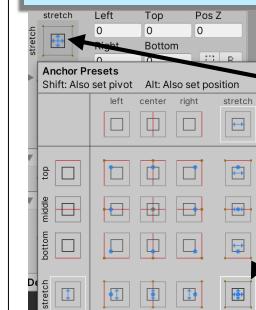
Our last two items are sound effects/Music and a Start and Game Over Screen.

These will require some classes that manage the state of our game.

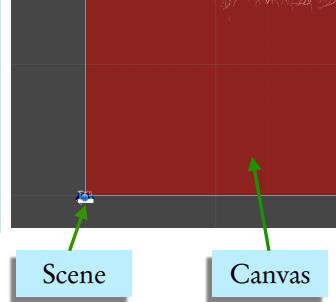
Wentworth
Computing & Data Science

Game Over Screen

In your scene view, the canvas will look huge. Don't worry, it will display properly when you run the game. However, we'll have to mess with the anchor settings to get it to look how we want.



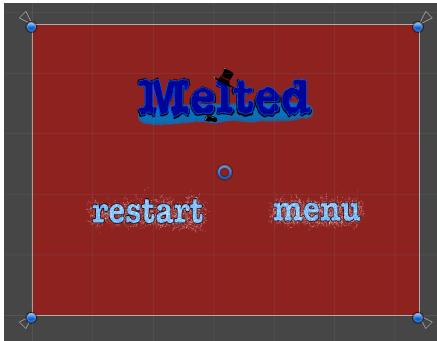
Click here to see the Anchor Presets



Notice that I have the stretch selected (in both dimensions). Using modifier keys (shift and alt) will set different anchor properties. Add more images or text and play around with the settings.

Game Over Screen

Here's an example:



'Melted' is just an image, but 'Restart' and 'Menu' contain an image and a button component. It's easy to add more components to your UI elements.



All of the objects are children to the panel that contains the background color (which itself is a child of the canvas).

Science

Game Over Screen

Here's an example:



'Melted' is just an image, but 'Restart' and 'Menu' contain an image and a button component. It's easy to add more components to your UI elements.

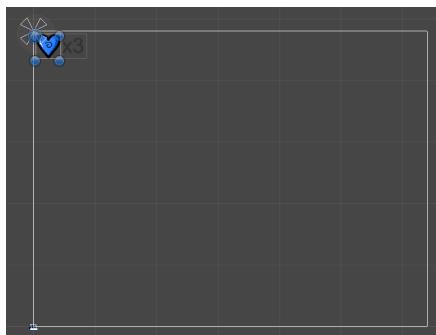


All of the objects are children to the panel that contains the background color (which itself is a child of the canvas).

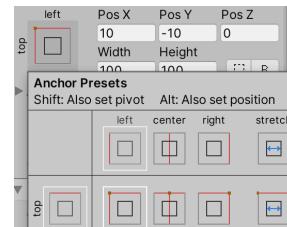
Science

Health UI

Here's the Health UI on the same Canvas:

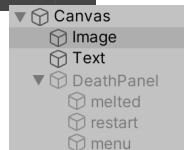


Note that the anchor is in the top left:



We'll update the text via code, for now I've entered "x3" as the default.

And the Image and Text are children of the Canvas (not the DeathPanel):



Remember: public fields are updated in the inspector.

```
public Text healthText;
public GameObject DeathPanel;

public void updateHealthText(int health){
    healthText.text = "x" + health
}

public void deathPanelSwitch(bool state){
    deathPanel.SetActive(state)
}
```

GameManager.cs

These are the public methods that allow us to update the GUI information, i.e. show the DeathPanel and change the text for the health.

Code Hookup

Now we can add the previous calls to the appropriate places in our current code:

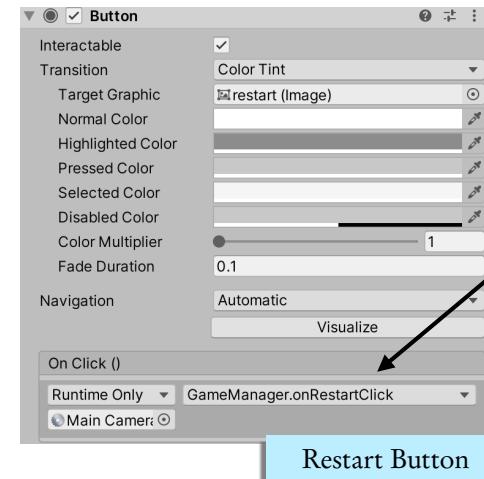
```
void Start(){  
    deathPanelSwitch(false);  
    ...  
}  
  
public void onRestartClick(){  
    if(deathPanel.activeSelf){  
        deathPanelSwitch(false);  
        player.reset();  
        spawner.reset()  
    }  
}
```

And create the method to call when the restart button is pressed

Wentworth
Computing & Data Science

GameManager.cs

Code Hookup



Restart Button

Wentworth
Computing & Data Science

Back to Player and Spawner

Our onRestartClick() method resets the player and the spawner:

```
public void reset(){  
    health=3;  
    this.transform.position=new Vector3(0f,0f,0f);  
    GameManager.instance().updateHealthText(health);  
    this.gameObject.SetActive(true)  
}
```

Position the player wherever you want, and ensure that the player gameObject is active.

Player.cs

```
public void reset(){  
    timeBetweenSpawns=1.25f;  
}
```

Spawner.cs

Reset spawner timing.

Wentworth
Computing & Data Science

Back to Player and Spawner

When the player takes damage and/or “dies” we need to make sure we trigger the GameOver screen

```
public void takeDamage(int value){  
    health-=value;  
    GameManager.instance().updateHealthText(health);  
    if(health<=0){  
        //Instantiate death effect here if you want  
        this.gameObject.SetActive(false);  
        GameManager.instance().deathPanelSwitch(true);  
    }  
}
```

This will make the player gameObject disappear (without destroying it in memory) and show the “DeathPanel”

Wentworth
Computing & Data Science

Lots of Little Stuff

OK! There's always lots of little questions that we have to answer, for example:

Q: Do we want fireballs to continue spawning after the players dies?

A: Add a guard clause in the spawner script update() to check if the player is active.

Q: How do we ensure the health is correct and the DeathPanel is off at the start of the game?

A: In the GameManager start(), call reset() on the player and spawner and call deathPanelSwitch(false)

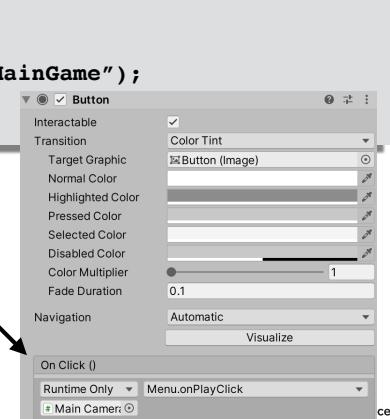
Wentworth
Computing & Data Science

Main Menu

Create a new script called Menu.cs and attach it to the Main Camera

```
public class Menu : MonoBehaviour
{
    public void onPlayClick(){
        SceneManager.LoadScene("MainGame");
    }
}
```

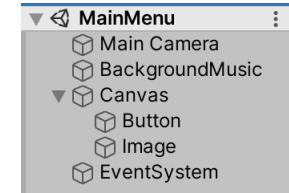
Hook this up to the button that you just created. This loads the scene "MainGame" and unloads the current scene.



Main Menu

Create a new Scene, call it MainMenu and put it into the scenes folder. Load this scene (double click it).

Ensure that there is a Camera in the scene and add a UI button and UI Image.



We'll add that BackgroundMusic gameobject in a moment
Wentworth
Computing & Data Science

Back in GameManager and the MainGame Scene

Create a method to 'fire' if the user presses the menu button from the GameOver screen:

```
public void onMenuClick(){
    if(deathPanel.activeSelf){
        SceneManager.LoadScene("MainMenu");
    }
}
```

Hook this up to the menu button on the DeathPanel.

This is how you change scenes.

Scenes can represent many things: Menus, levels, cutscenes, etc.

Wentworth
Computing & Data Science

Music and Sound

For now, we're just going to add the background music. Adding sound effects will be up to you in the homework.

We'll add the new gameObjects to the MainMenu scene.

Then, we'll set it to persist across scene transitions.

Sound effects will work similarly to what you see here, though they will likely just play once.

Feel free to use whatever software you want for creating music and sound effects.

Music:

- Bosca Ceoil
- LMMS

SoundFX:

- Audacity + your voice
- Bfxr

Wentworth
Computing & Data Science

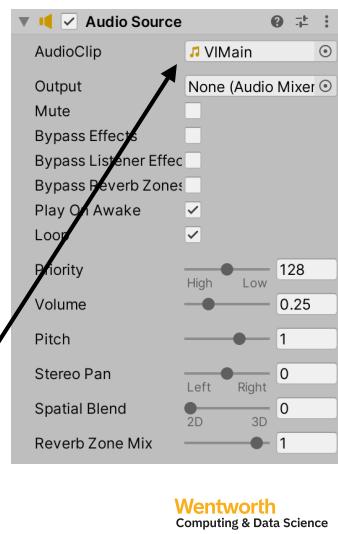
Background Music

Let's start by creating a new empty gameobject, name it "Background Music" and add the "Audio Source" component.

Make sure to have Play on Awake and Loop enabled.

Now create a new script, MusicManager and add it to the gameobject that we just created.

Since this will be our background music, drag your audio clip here



Wentworth
Computing & Data Science

Background Music

We'll start by making the MusicManager class a singleton and set it so that changing scenes doesn't destroy the object (so the music keeps playing).

```
[RequireComponent(typeof(AudioSource))]  
public class MusicManager : MonoBehaviour  
{  
    AudioClip backgroundClip;  
  
    static MusicManager _instance=null;  
  
    void Awake()  
    {  
        if(_instance==null){  
            _instance=this;  
        } else {  
            Destroy(this.gameObject);  
        }  
        DontDestroyOnLoad(this.gameObject);  
    }  
}
```

MusicManager Class

Background Music

The start method is what grabs a reference to the AudioSource component and sets the backgroundClip. Then, if the clip is not playing, play it.

```
void Start()  
{  
    AudioSource source=GetComponent<    backgroundClip=source.clip;  
  
    if(!source.isPlaying){  
        source.Play();  
    }  
}
```

MusicManager Class

Wentworth
Computing & Data Science

Coming Up:

- Shader Graph