GUIs, Menus & Particle Systems

Introduction

Unity UI System

A good system should adequately display information to a player

The Heads-Up Display HUD should be a transparent display used to show information a player would need to know about during game play

A HUD should not have information overload, and typically it is better to show elements in the corners of the screen

Information should not be too little or too much

Visual Components

20 Minutes Till Dawn

Legend of Zelda: Breath of Wild

Apex Legends

Devil May Cry 5

Horizon Zero Dawn: Forbidden West

Text

This is for non-interactive text (i.e., timer, score, combo count)

Image

UI images also (i.e, bullet icons, hearts/health bar)

UI images should be imported as a sprite

Button

UI button comes with an image (that can be reset by a developer)

UI button common options:

- Interactable:
- Transition:
- Navigation: Allows to specify how to move between UI elements using arrow keys (i.e. move to audio menu after clicking audio in the menu and likewise for the sound, gameplay, etc.)
- OnClick(): A list of functions that can be will the button is clicked

Slider

Usage examples:

- Audio control
- Showing audio input
- Brightness control
- Health bar

The area that holds UI elements

All UI elements are children of the Canvas

Later elements will be displayed over earlier ones (based on the hierarchy)

Render mode of a Canvas can determine how elements are displayed over the game (i.e., always on top, a certain distance from camera, as a game object, etc.)

Scaling is not recommended for UI elements and Resizing is better

Scaling example: Doubling the overall size of the image

Resizing example: Change font size from 12 to 18

Scaling is good for animated and special effects for UI elements

Negative scaling can be used to flip elements, whereas negative sizing will making the element invisible

Pivot is the location of the pivot point around the rectangle rotates, and is defined as a fraction of the size of the rectangle itself (0 to 1)

UI Events & Triggers

Events can be attached to game objects are their properties (i.e. player -> health property for health bar)

The Event System can be used to trigger events (via OnClick, etc.)

Menus and Loading Scenes

Fade In and Out

Examples:

- Background sunset
- Changing scene
- Item drop expires

{See lecture slides 27 - 29 for this script}

Changing Scenes

LoadScene can be given a build index or a scene name

To be able to use a scene it must be added to the list of scenes in the build order (File → Build Settings → Build order)

When a new scene is loaded the current scene is destroyed by default

Additive scenes can be used to maintain information between scenes

Additive scenes are quite costly and should be optimized (i.e., don't copy all information you want to keep between scenes, also which scenes are presented/render based on distance)

A level select can be placed as an independent scene

Loading Scenes

A loading scene

To add a loading screen, we can add an image with a black background colour and a text that indicates that the scene is loading. Set as a public variable of the same script, we can just set it to active when the event is captured

```
public GameObject loadingImage;

public void LoadScene(int leveldx)
{
    loadingImage.SetActive(true);
    SceneManager.LoadScene(leveldx);
}
```

A delegate can be added to the SceneManager.sceneLoaded event to detect when a scene has been loaded

A delegate is a function that captures an event triggered by Unity

Loading Scenes Additively

Pausing

- External interruptions
- Accessing the menu

TimeScale

This is the scale at which in-game time is running

1.0 is normal time, 0.5 would be x2 slower, 2 would be x2 faster, 0 pause the game (if all the functions are frame rate independent)

It is recommended to also lower Time.fixedDeltaTime is you are lowering Time.timeScale

Canvas

Particles

Billboarding renders a 3D model as a 2D place

Billboarding is good for rendering far away elements

Billboarding draws 2D textures onto 3D rectangles (quads) place in the scene. These quads will typically always been rotated so that they are always facing the camera in order to create the illusion of a 3D model at a much lower cost

Sprites (2D) and Meshes (3D) are good for depicting *solid* game objects

Particles are good for non-solid objects

Particles are composed of a collection of billboards, and so have a texture

Particles are entirely visual and very small and used to increase immersion

Examples include dust, rain, water, flames, explosions, smoke (i.e. death animation)

Each particle is:

- Rotated towards the camera
- Starts at a random position, within a region (sphere, con, box, defined mesh, etc.)
- Has a lifetime
- Has a velocity vector for direction and distance moved each frame

The Particle System Component

The emission rate of a particle system denotes how many "particles" are generated per second

Particles are typically faded in (creation) and out (destruction) for smooth transitions

Common global settings for this are: duration, looping, start delay, lifetime, etc.

Bursts can be added to create "bursts" of additional particles that appear at a certain time

Shaped module can be used to define the shape of the area particles are launched from (i.e., con, sphere, box)

Collisions module defines the way collisions between solid objects in the scene are dealt with (i.e., a ceiling, platform over the particle system)

Particle Collisions

Summary

Quiz