UI 23-Comic Menu

Details Description

1. Basic Description

The Comic Art UI has been designed with a comic book aesthetic, featuring bold lines, bright colors, and a stylized free font. The Comic Game Art UI project is a unique addition to the Unity Asset Store, offering developers a fun and playful way to enhance the look and feel of their games. The UI includes various elements, such as buttons, menus, and icons, all designed to work together seamlessly. Whether you're creating a superhero adventure or a cartoon-style puzzle game, the Comic Game Art UI will surely add a touch of whimsy and charm to your project.

2. Technical Description

The Comic Menu in UI 23 was created using the free Unity asset "Shapes2D." With Shapes2D, you can easily generate shader-based procedural sprites and UI elements like rectangles, triangles, lines, and ellipses using the Unity inspector. The inspector allows for extensive customization with fills, gradients, outlines, and other features. This project also relies on another Unity-free asset, "DOTween," to animate the sprites and UI items.

The links to these third-party plugins are given below:

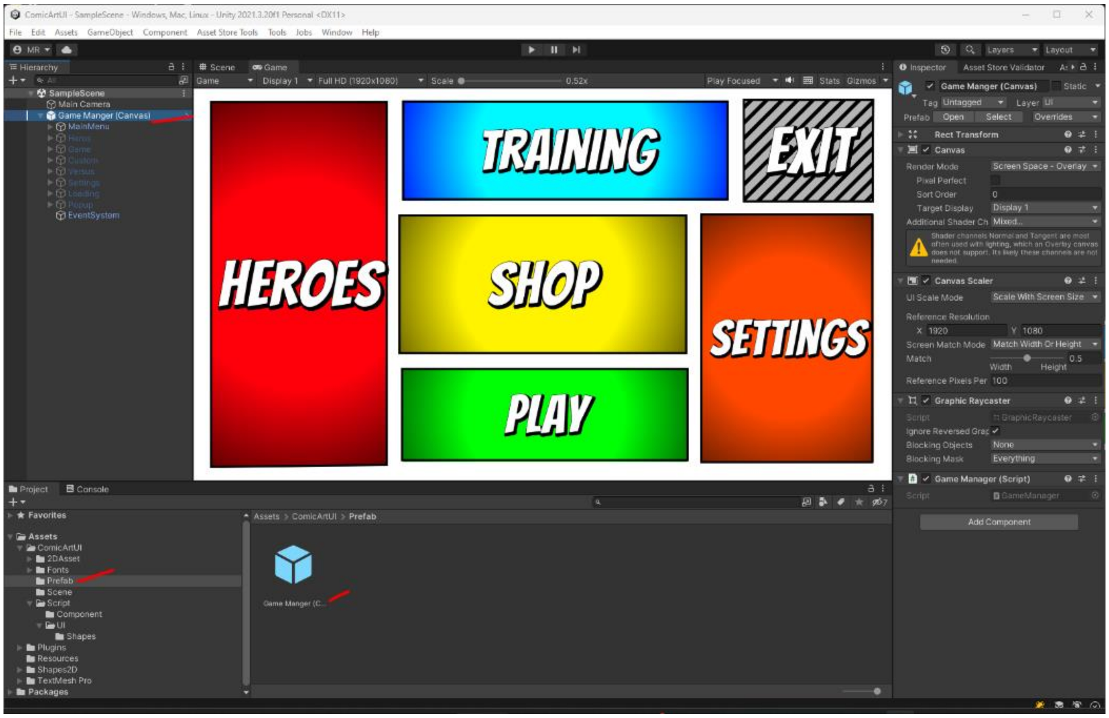
DOTween (HOTween v2): DOTween (HOTween v2) | Animation Tools | Unity Asset Store

Shapes2D – Procedural sprites and UI: Shapes2D - Procedural sprites and UI | Sprite Management | Unity Asset Store

3. How to use this project

3.1 Play

Download the project latest version from the Unity Asset Store. To use this project, drag the Game Manager prefab from the project prefab folder into your game scene shown in Figure 1 and press play.



ComicArtUI - SampleScene - Windows, Mac, Linux - Unity 2021.3.20f1 Personal <DX11>

Game Manger (Canvas)
Tag Untagged
Layer UI

Canvas
Render Mode Screen Space - Overlay
Pixel Perfect
Sort Order 0
Target Display Display 1
Additional Shader Ch Mixed...

Shader channels Normal and Tangent are most often used with lighting, which an Overlay canvas does not support. Its likely these channels are not needed.

Canvas Scaler
UI Scale Mode Scale With Screen Size
Reference Resolution X 1920 Y 1080
Screen Match Mode Match Width Or Height
Match 0.5
Reference Pixels Per 100

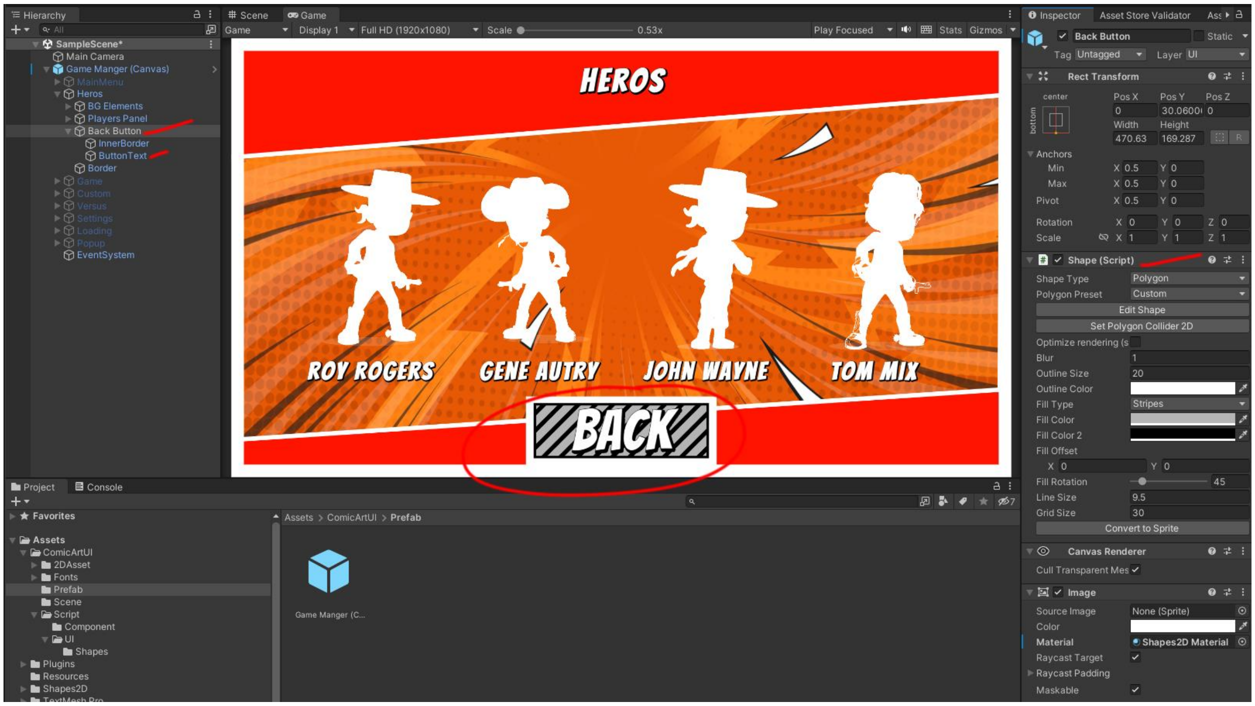
Graphic Raycaster
Script GraphicRaycaster
Ignore Reversed Grap ✓
Blocking Objects None
Blocking Mask Everything

Game Manager (Script)
Script GameManager

Figure 1

3.2 Text & Color

This project is based on Unity UI, where the Game Manager is a UI canvas-type object. To create you own button create an UI Image in canvas, then add a component named Shape.cs script to manipulate your image. Add a child UI Text component to this Image to visible label like a button. You can conveniently customize everything using the Unity inspector. In Figure 2 we mention a visible example, here you can see that Back-Button background image is created using the Shape.cs script. Now you can modify this button color stipes Shape Fill Color and Fill Color 2. Here you can easily change the button text.



Back Button — Static
Tag: Untagged — Layer: UI

Rect Transform
center / bottom
Pos X: 0 — Pos Y: 30.0600(— Pos Z: 0
Width: 470.63 — Height: 169.287
Anchors
Min: X 0.5 Y 0
Max: X 0.5 Y 0
Pivot: X 0.5 Y 0
Rotation: X 0 Y 0 Z 0
Scale: X 1 Y 1 Z 1

Shape (Script)
Shape Type: Polygon
Polygon Preset: Custom
Edit Shape
Set Polygon Collider 2D
Optimize rendering (s: □
Blur: 1
Outline Size: 20
Outline Color
Fill Type: Stripes
Fill Color
Fill Color 2
Fill Offset: X 0 Y 0
Fill Rotation: 45
Line Size: 9.5
Grid Size: 30
Convert to Sprite

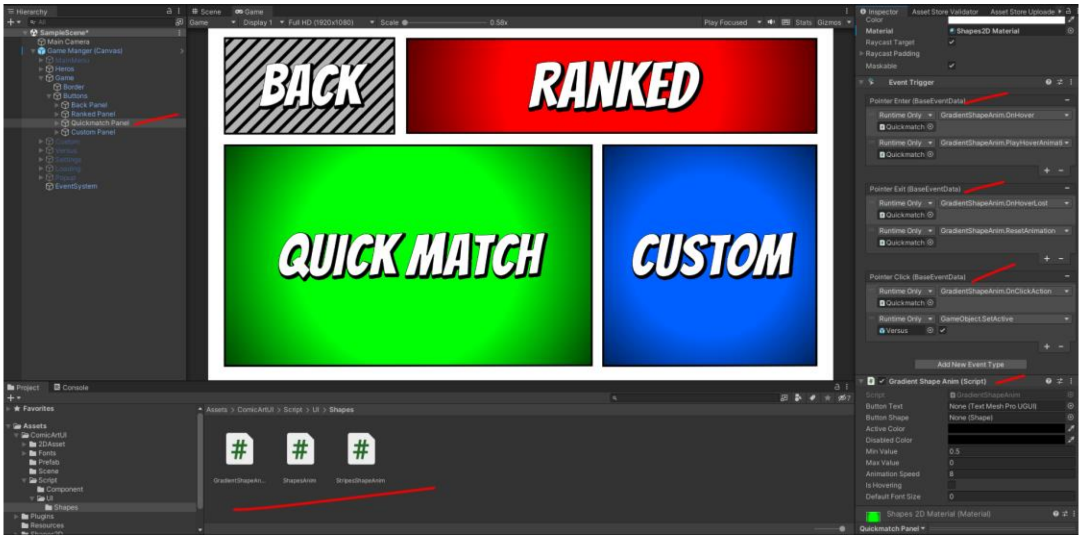
Canvas Renderer
Cull Transparent Mes: ✓

Image
Source Image: None (Sprite)
Color
Material: Shapes2D Material
Raycast Target: ✓
Raycast Padding
Maskable: ✓

Figure 2

3.3 Animation

You can easily create your button with animation, to do this add **Event Trigger** component and **GradientShapeAnim.cs** to your button, then assign **OnHover**, **PlayHoverAnimation** on Pointer Enter event, **OnHoverLost**, **ResetAnimation** on Pointer Exit event & **OnClickAction** on Point Click event as mention in Figure 3



Color
Material: Shapes2D Material
Raycast Target: ✓
Raycast Padding
Maskable: ✓

Event Trigger

Pointer Enter (BaseEventData)
Runtime Only — GradientShapeAnim.OnHover — Quickmatch
Runtime Only — GradientShapeAnim.PlayHoverAnimati — Quickmatch

Pointer Exit (BaseEventData)
Runtime Only — GradientShapeAnim.OnHoverLost — Quickmatch
Runtime Only — GradientShapeAnim.ResetAnimation — Quickmatch

Pointer Click (BaseEventData)
Runtime Only — GradientShapeAnim.OnClickAction — Quickmatch
Runtime Only — GameObject.SetActive — Versus ✓

Add New Event Type

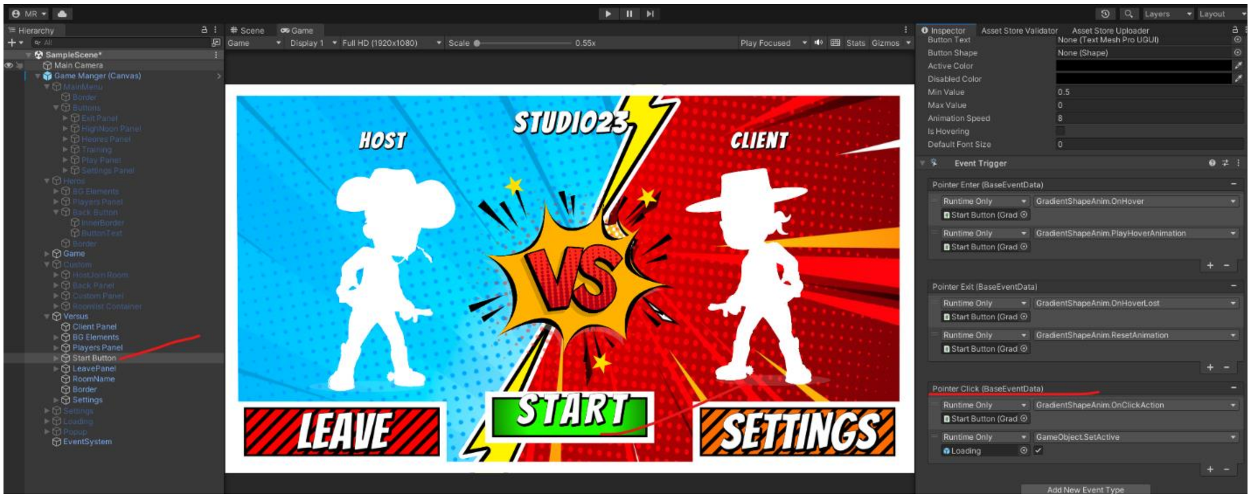
Gradient Shape Anim (Script)
Script: GradientShapeAnim
Button Text: None (Text Mesh Pro UGUI)
Button Shape: None (Shape)
Active Color
Disabled Color
Min Value: 0.5
Max Value: 0
Animation Speed: 8
Is Hovering: □
Default Font Size: 0

Shapes 2D Material (Material)
Quickmatch Panel

Figure 3

3.4 Button click event

You can change what happened when a button is clicked from Event Trigger **Pointer Click** shown in figure 4



Button Text: None (Text Mesh Pro UGUI)
Button Shape: None (Shape)
Active Color
Disabled Color
Min Value: 0.5
Max Value: 0
Animation Speed: 8
Is Hovering: □
Default Font Size: 0

Event Trigger

Pointer Enter (BaseEventData)
- Runtime Only — GradientShapeAnim.OnHover — Start Button (Grad
- Runtime Only — GradientShapeAnim.PlayHoverAnimation — Start Button (Grad

Pointer Exit (BaseEventData)
- Runtime Only — GradientShapeAnim.OnHoverLost — Start Button (Grad
- Runtime Only — GradientShapeAnim.ResetAnimation — Start Button (Grad

Pointer Click (BaseEventData)
- Runtime Only — GradientShapeAnim.OnClickAction — Start Button (Grad
- Runtime Only — GameObject.SetActive — Loading ☑

Figure 4